TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

KALLE KOIVISTO

LUOTETUN SISÄLTÖÄ SEULOVAN DIGITAALISEN ALLEKIRJOITUSJÄRJESTELMÄN TOTEUTUS

Diplomityö

Tarkastajat: Tommi Mikkonen (TTY)
Antti Stenhäll (Intel)
Jiri Uitto
Tarkastaja ja aihe hyväksytty osastoneuvoston kokouksessa 3.6.2015

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma
KALLE KOIVISTO: Luotetun sisältöä seulovan digitaalisen allekirjoitusjärjestelmän toteutus
Diplomityö, 58 sivua
Kesäkuu 2016
Pääaine: Sulautetut järjestelmät
Tarkastajat: Professori Tommi Mikkonen
Avainsanat: digitaalinen allekirjoitus, tietoturva

Digitaalinen allekirjoittaminen on yksi yrityksen tietoturvan kulmakivistä. Käsin tehdyn allekirjoituksen tavoin digitaalisen allekirjoituksen tuottaja menee takuuseen allekirjoittamansa tiedon sisällöstä.

Tässä työssä käydään läpi Intelille suunniteltu ja toteutettu digitaalinen allekirjoitusjärjestelmä, joka kykenee seulomaan yhtiön tuotteiden tietoturvan kannalta tärkeitä konfiguraatiopaketteja ennen allekirjoittamista. Seulonta tapahtuu allekirjoitusprosessin yhteydessä keskitetyillä allekirjoituspalvelimilla.

Järjestelmän ensisijainen tavoite on täyttää Intelin mobiililaitteiden tuotekehitykseen liittyvät allekirjoitusvaatimukset, mutta sen on tarkoitus kyetä skaalautumaan mahdollisimman pienellä vaivalla myös muihin yhtiön allekirjoitusvaatimuksiin.

Luotettavuuden kannalta oleellista on, että järjestelmässä käytetyt allekirjoitusavaimet on suojattu HSM-kryptolaitteiden (engl. hardware security module) avulla ja järjestelmän vaatimukset on linjattu yhdessä yhtiön tietoturva-asiantuntijoiden kanssa.

Itse seulonta tapahtuu Lua-skripteillä, jotka ajetaan allekirjoituspalvelimien yhteyteen sulautetulla Lua-tulkilla. Seulontaskriptit hyödyntävät sisäisiä funktiorajapintoja allekirjoitettavan tiedon lukemiseen ja siihen kirjoittamiseen.

Toteutusta voidaan pitää onnistuneena, sillä järjestelmä hyväksyttiin Intelin tuotantokäyttöön ja sen katsottiin täyttävän sille asetetut vaatimukset. Tuotantokäytössä järjestelmässä ei ole ilmennyt merkittäviä puutteita tai ongelmia. Jatkokehitysideoita ja -mahdollisuuksia kuitenkin on.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Masters's Degree Programme in Electrical Engineering
KALLE KOIVISTO: Design and implementation of a trusted screening-capable digital signing service
Master of Science Thesis, 58 pages
June 2016
Major: Embedded Systems
Examiners: Professor Tommi Mikkonen
Keywords: digital signing, information security

Digital signing is a trust anchor in corporate level information security. Just like in the case of classical hand-written signature, the producer of digital signature approves the data content he or she is signing.

This thesis introduces the design and implementation of a trusted digital signing service that is capable to do automated screening for the input data. The screening takes place on centralized signing servers before the signing operation.

The primary purpose of the system is to meet requirements and needs set by Intel's mobile device RnD units but it is meant to be scalable to other signing scheme requirements of the company.

To gain trust, it is crucial that the keys are encrypted and protected using Hardware Security Modules (HSM) and the requirements are gathered together with the security experts at Intel.

The screening practically happens through Lua scripts which are executed in a Lua engine inside the signing server. The scripts have access to internal API functions which allow them to read and write data that is being signed.

The implementation can be considered to be successful. The implemented system was approved to production use and it was agreed to fulfil the set requirements. There have not been any major issues in the production use but there are potential ideas and possitilibites for the further development of the implementation.

ALKUSANAT

Olen kirjoittanut tämän diplomityön palkkatyöni ohella työskennellessäni Intelillä. Haluan lämpimästi kiittää kaikkia kyseisessä projektissa auttaneita. Erityisesti haluan kiittää Jiri Uittoa ensinnäkin aiheen tarjoamisesta mutta myös minuun uskomisesta ja tukena olemisesta. Myös professori Tommi Mikkonen ansaitsee suuret kiitokset kannustavasta ja rakentavasta palautteestaan. Projektin osalta haluan kiittää Laura Warneria projektiin liittyvien papereiden ja hallinnoinnin hoitamisesta. Työyhteisö Tampereen toimistolla on ollut myös kannustava, ja haluan antaa erityiskiitokset Antti Stenhällille, Brian McGillionille, Rauno Tammiselle ja Laura Moisiolle vinkeistä ja palautteesta. Lopuksi haluan kiittää läheisiäni, jotka ovat jatkuvasti kannustaneet minua tämän työn loppuun saattamiseksi.

Tampereella 19.5.2016

SISÄLTÖ

1. JOHDANTO

Allekirjoitusjärjestelmät ovat yksi digitaalisen ja verkottuneen maailman tietoturvan kulmakivistä. Nykyään lähes koko internetin salattu liikenne perustuu digitaalisesti allekirjoitettuihin varmenteisiin. Varmenteiden lisäksi yleinen esimerkki on ohjelmistot suosituille mobiili- sekä työpöytäkäyttöjärjestelmille. Luottamus näihin ohjelmistoihin, kuten esimerkiksi mobiilipeleihin, perustuu valmistajien tuotteidensa mukana toimittamiin allekirjoituksiin. Allekirjoittamalla tietoa allekirjoittaja hyväksyy ja menee takuuseen sen sisällöstä. Useissa maissa, kuten Yhdysvalloissa ja Euroopan unionin jäsenmaissa, digitaalisella allekirjoituksella on juridinen merkitys. Esimerkiksi Suomessa digitaalinen allekirjoitus on verrattavissa käsin kirjoitettuun allekirjoitukseen [1]. Tästä syystä virheelliset allekirjoitukset voivat olla erittäin vahingollisia sekä yrityksen omalle että asiakkaiden liiketoiminnalle ja imagolle. Digitaalisten allekirjoitusten tuottaminen jonkun toisen nimissä on monesta syystä houkutteleva kohde kyberrikollisille.

Tässä diplomityössä on suunniteltu ja toteutettu luotettu sisältöä seulova digitaalinen allekirjoitusjärjestelmä. Projekti alkoi prototyyppiversiosta ja kehittyi sittemmin tuotantoversioksi. Tuotantoversion ohjelmointityössä käytettiin osittain alihankintaa projektin nopeuttamiseksi. Järjestelmä on suunniteltu ensisijaisesti Intelin mobiilialustojen ja niihin liittyvien ohjelmisto- ja varusohjelmakomponenttien allekirjoittamiseen, mutta suunnittelussa on huomioitu muidenkin Intelin yksiköiden mahdollisia allekirjoitustarpeita. Seulonnalla tarkoitetaan menettelyä, jossa allekirjoituspalvelimet lukevat automatisoidusti allekirjoitettavan tiedon kriittiset kohdat läpi ennen varsinaista allekirjoittamista. Seulontaa voikin verrata perinteiseen allekirjoitukseen niin, että allekirjoittava henkilö lukee sopimuksen tai muun dokumentin huolellisesti läpi ennen sen hyväksymistä omalla allekirjoituksellaan. Ensisijaisesti seulonnalla pyritään ratkaisemaan Intelillä esiintynyt ongelma, jossa allekirjoitusjärjestelmät allekirjoittivat pelkkiä digitaalisen tiedon hajautusarvoja (engl. hash) tietämättä alkuperäistä sisältöä. Seulonnan automatisoinnilla on myös mahdollista säästää henkilöresursseja sekä lisätä järjestelmän luotettavuutta pienentämäl-

lä inhimillisen virheen mahdollisuutta. Luotettavuudella viitataan siihen, että toteutetun järjestelmän tuottamien allekirjoitusten alkuperä on yksiselitteinen.

Luvussa kaksi käsitellään digitaalisen allekirjoituksen taustoja ja teoriaa selittäen muun muassa periaatteita sekä yleisiä algoritmeja. Myös digitaalisen allekirjoitusjärjestelmän turvallisuustekijöitä, joka on erittäin olennainen osa koko järjestelmää, käydään läpi tässä luvussa. Lähtökohtaisesti allekirjoitusjärjestelmä on yksi tietoturvan kulmakivistä ja sen täytyy tästä syystä olla luotettu. Tässä toteutuksessa ankkureina (engl. trust anchor) on käytetty HSM-kryptolaitteita (engl. Hardware Security Module).

Kolmannessa luvussa esitellään varsinainen toteutus. Alussa käydään läpi järjestelmälle asetetut vaatimukset sekä suunnitteluperiaatteet. Lisäksi esitellään toteutukseen kuuluneet osat, kuten erinäiset sovellukset ja mallit, sekä erinäiset järjestelmäkuvaukset. Vaatimukset syntyivät Intelin omien tietoturva-asiantuntijoiden, mikroarkkitehtien ja tuotekehitysyksiköiden kesken. Suuri osa näistä ihmisistä työskenteli Suomessa, mikä helpotti iterointia läpi projektin.

Neljännessä luvussa arvioidaan toteutettua järjestelmää eri näkökulmista, kuten esimerkiksi vaatimusten täyttyminen, suorituskyky, tietoturva ja käytettävyys. Arviointi- ja testitulosten sekä löydettyjen ongelmien perusteella luvun lopussa on myös esitetty jatkokehitysajatuksia.

Luku viisi on työn yhteenveto. Tässä luvussa kerrataan vielä lyhyesti, mitä pyrittiin tekemään ja mitä siitä saavutettiin.

2. TAUSTOJA

Tämän luvun kohta 2.1. käy läpi syitä, miksi luotettu allekirjoitusjärjestelmä on yritykselle tärkeä tietoturvatekijä. Kohta 2.2. esittelee digitaalisen allekirjoittamisen teoriaa ja taustoja. Esittelyssä on muun muassa yleisperiaate sekä allekirjoitusjärjestelmään liittyviä algoritmeja. Kohta 2.3. sisältää yleisesti tietoturvaan ja tietoturvasuunnitteluun liittyviä periaatteita, joihin tämän työn suunnittelu nojaa. Läpi käytävät asiat ovat yhteisiä lähes kaikille teollisuudessa tehtäville tietoturvakeskeisille ohjelmistoprojekteille.

2.1. Tarve luotetuille allekirjoituksille

Digitaalisen allekirjoituksen käyttötarkoitus on verrattavissa tavalliseen käsin tehtyyn allekirjoitukseen: Allekirjoittava taho todistaa vastaanottajalle, että hän on luonut kyseisen viestin tai hyväksynyt sen sisällön. Digitaalisen allekirjoituksen tapauksessa allekirjoittaja voi olla luonnollisen henkilön lisäksi myös laite tai ohjelmisto, esimerkiksi allekirjoituspalvelin.

Luotettavan allekirjoituksen tuottamiseksi järjestelmän on tiedettävä, mitä tietoa se allekirjoittaa. Yksinkertaisissa allekirjoitusjärjestelmissä allekirjoitetaan pelkkä hajautusarvo (engl. hash) alkuperäisestä tiedosta. Koska vahvalla algoritmilla tuotetusta hajautusarvosta ei voida päätellä alkuperäisen tiedon sisältöä, ei allekirjoitusjärjestelmä pysty todentamaan sen allekirjoittamaa tietoa. Tällaisen allekirjoituksen tuottaminen on riskialtista, sillä se perustuu täysin luottamukseen tiedon lähettäjän ja allekirjoitusjärjestelmän välillä. Tätä ongelmaa pyritään tässä työssä ratkaisemaan allekirjoitettavan tiedon seulonnalla.

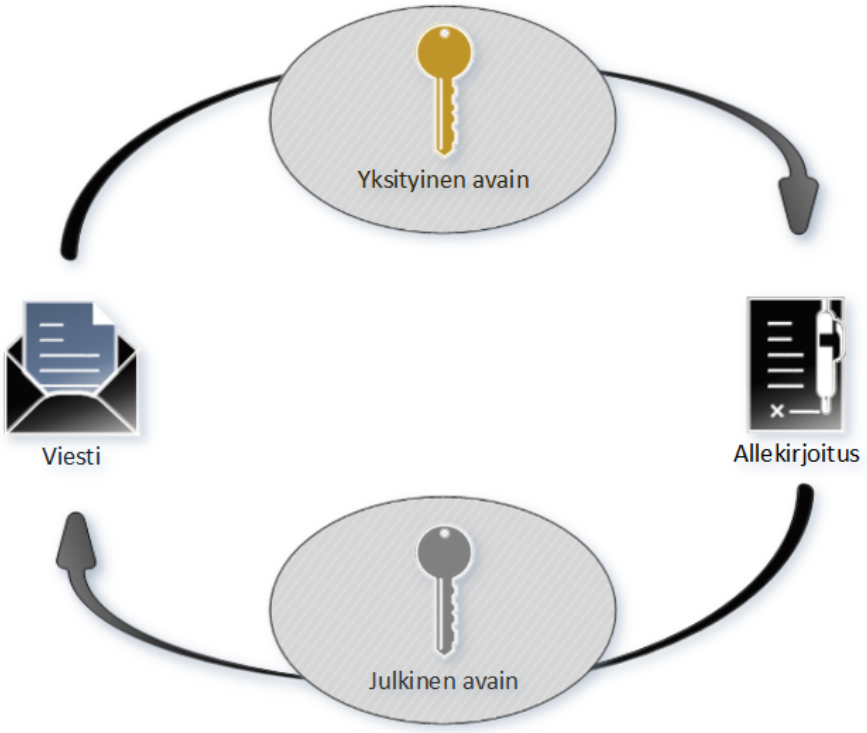
Koska allekirjoitusjärjestelmä kokonaisuudessaan luo perustan allekirjoitetun viestin tai asian luotettavuudelle, on tärkeää, että itse allekirjoitusjärjestelmä ja sen käyttämä prosessi on asianmukaisesti suojattu. Jos allekirjoitusjärjestelmä on haavoittuva eikä siihen voida luottaa, ei myöskään sillä allekirjoitettuihin viesteihin voida luottaa. Murrettu allekirjoitusjärjestelmä voi aiheuttaa omistajalleen valtavat taloudelliset vahingot ja tehdä

haittaa imagolle ja maineelle. Se voi myös vaarantaa yksittäisten ihmisten tietoturvan. Yksi esimerkki on matkapuhelinvalmistaja Nokialle aiheutuneet kustannukset murretusta allekirjoitusjärjestelmästä. Vuodenvaihteen 2007 ja 2008 tienoilla Nokia joutui maksamaan miljoonia euroja kiristäjille, jotka olivat väitetysti saaneet haltuunsa Nokian Symbian-järjestelmissä käyttämiä allekirjoitusavaimia [2]. Tapaus on tätä työtä tehdessä edelleen ratkaisematta, mikä kertoo tietomurtojen jäljitettävyyden haasteellisuudesta.

2.2. Digitaalisen allekirjoittamisen yleisperiaatteet

Digitaaliset allekirjoitusjärjestelmät hyödyntävät usein asymmetrista kryptografiaa. Tästä käytetään nimeä julkisen avaimen menetelmä (PKI), jota kuva 2.1 havainnollistaa. Julkisen avaimen menetelmä esiteltiin jo vuonna 1976 ja siinä pääperiaatteita on kaksi [3]:

- Tuotetun allekirjoituksen alkuperä voidaan todentaa allekirjoituksessa käytetyn avainparin julkisen avaimen avulla.
- Julkisen avaimen avulla ei saa pystyä tuottamaan vastaavaa allekirjoitusta eikä selvittämään avainparin yksityistä avainta.



Kuva 2.1: Digitaalisen allekirjoituksen perusmalli

Digitaalisen allekirjoituksen pääajatus on, että yhdenkin bitin muuttuessa alkuperäisessä tiedossa siitä tuotettu allekirjoitus on erilainen. Tyypillisesti allekirjoitusjärjestelmään kuuluu kolme pääalgoritmia:

- Avaimenluontialgoritmi: Luo avainparin eli privaattiavaimen ja julkisen avaimen.
- Allekirjoitusalgoritmi: Tuottaa allekirjoitettavan tiedon ja privaattiavaimen avulla itse allekirjoituksen.
- Varmennusalgoritmi: Hyväksyy tai hylkää allekirjoituksen käyttäen allekirjoittajan julkista avainta.

Näiden lisäksi käytetään usein algoritmeja laskemaan tiedon hajautusarvoja. Hajautusarvojen avulla pystytään esittämään tietoa alkuperäistä pienemmässä koossa. Hajautusarvoja käytetäänkin lähinnä käytännöllisistä syistä ja allekirjoitusjärjestelmissä mahdollisia käyttökohteita on monia (kts. alakohta 2.2.3.).

Ensimmäinen varteenotettava käytännön toteutus digitaalisesta allekirjoittamisesta oli vuonna 1977 esitelty RSA-algoritmi, joka mahdollisti yksinkertaisten allekirjoitusten tuottamisen [4]. Algoritmin nimi on lyhenne keksijöidensä Ronald Rivestin, Adi Shamirin ja Len Adlemanin sukunimien alkukirjaimista. Nykyään RSA-nimeä kantava yhtiö kehittää allekirjoitus- ja salausalgoritmejä, jotka ovat maailmanlaajuisesti suosittuja.

2.2.1. Avaimenluontialgoritmit

Tyypillisesti avaimen luonnin perustana käytetään syötettä, joka toimii sisääntulona salausfunktiolle. Modernit kryptografiset järjestelmät hyödyntävät avaimenluonnissa symmetrisiä salausalgoritmeja, joita on listattu taulukossa 2.1. Avain itsessään on suuri kokonaisluku, joka rajoittuu avaimelle asetettuun maksimikokoon.

Esimerkiksi Yhdysvaltojen kansallisen standardi- ja teknologiainstituutin (NIST) luotettavana pitämien RSA-avainten minimikoko on toistaiseksi kaksi kilobittiä [9, s. 6], mutta avaimen kokoa valittaessa täytyy ottaa huomioon avaimen käyttöikä. Teknologian ja laskentatehon kehittyessä kustannukset avainten murtamiseen pienenevät. Avaimen koon kasvattaminen kasvattaa sen murtamiseen tarvittavaa aikaa ja laskentatehoa. Toisaalta isojen avainten käyttäminen on raskaampaa ja ne vievät enemmän tilaa. Tulevaisuudessa

kvanttitietokoneiden on povattu kykenevän murtamaan nykyiset kahden kilobitin RSA-avaimet niin nopeasti, että hyökkäyskustannusten pienentymisen johdosta lähes koko internetin salattu liikenne on vaarassa. Tämän vuoksi muun muassa NSA on julkistanut omat suosituksensa avainten suuruuksille ja tyypeille [8]. Tämä on RSA-avaimille kolme kilobittiä.

Taulukko 2.1: Salausalgoritmeja

Julkaistu	Nimi	Lyhenne
2001	Advanced Encryption Standard	AES
1998	Triple Data Encryption Algorithm	3DES
1976	Data Encryption Standard	DES

NIST on myös määritellyt AES-salaukselle luotettavat avainkoot, jotka ovat 128, 192 ja 256 bittiä. Vastaavasti 3DES-salauksessa suositellaan kolmen avaimen variaatiota. [9, s. 4]

2.2.2. Allekirjoitus- ja varmennusalgoritmit

Allekirjoitusalgoritmit hyödyntävät allekirjoitusavaimia ja tuottavat allekirjoituksia. Taulukossa 2.2 on listattu yleisiä allekirjoitusalgoritmeja. Yhdysvaltojen kansallinen standardi- ja teknologiainstituutti (NIST) on hyväksynyt FIPS 186-4 standardin, joka määrittelee luotetuiksi allekirjoitusalgoritmeiksi DSA:n, RSA:n ja ECDSA:n [5].

Taulukko 2.2: Allekirjoitusalgoritmeja [4] [6] [7]

Julkaistu	Nimi	Lyhenne
1999	Elliptic Curve Digital Signature Algorithm	ECDSA
1991	Digital Signature Algorithm	DSA
1984	ElGamal Signature Scheme	-
1977	RSA Signing Algorithm	RSA

RSA on yhtiönä kehittänyt allekirjoitusalgoritmiaan ja ylläpitää omaa PKCS-dokumenttijärjestelmäänsä (lyhenne sanoista Public Key Cryptography Standard), josta PKCS#1 esittelee käytettävät mekanismit RSA-pohjaiselle allekirjoittamiselle. Yhdysvalloissa NIST on hyväksynyt käytettäväksi PKCS#1:n versiossa 2.1 esitellyt algoritmit RSASSA-PKCS1 v1.5 ja RSASSA-PSS [5, s. 24-25].

2.2.3. Hajautusalgoritmit

Usein tiedosta lasketaan tiiviste eli hajautusarvo (engl. hash), jonka avulla alkuperäistä tietoa voidaan vertailla paljon pienemmässä muodossa. Hajautusarvon toiminta perustuu siihen, että sen perusteella ei voida palauttaa tiedon alkuperäistä arvoa. Hajautusarvo ei siis saa vaarantaa alkuperäistä tietoa, jos se halutaan pitää salassa. Hajautusarvoja lasketaan esimerkiksi allekirjoitusavaimista sekä allekirjoitettavasta tiedosta.

Yksi yleisimmistä algoritmeista nykypäivänä on NSA:n kehittämä SHA2-algoritmiperhe, jota myös NIST pitää luotettavana [9]. SHA2-algoritmiperheen edeltäjä oli SHA1, joka puolestaan korvasi aikaisemmin yleisen MD5-algoritmin. MD5-algoritmi todettiin käyttökelvottomaksi vuonna 2008 [10].

2.3. Tietoturvasuunnittelu

Tietoturva on aihealueena laaja. Sen vuoksi tässä kohdassa sen taustoittaminen rajataan toteutetun järjestelmän osalta kiinnostaviin aiheisiin. Näitä ovat yleisten periaatteiden lisäksi järjestelmälle potentiaaliset hyökkäyskohteet sekä -tavat.

2.3.1. Perusperiaatteet

Tietoturvan perinteiset peruselementit ovat luottamuksellisuus (engl. confidentiality), eheys (engl. integrity) ja saatavuus (engl. availability). Paremman kattavuuden vuoksi näitä on kuitenkin myöhemmin jatkettu kahdella elementillä: kiistämättömyys (engl. non-repudiation) ja todentaminen (engl. authentication). [11, s. 20-23] Nämä voidaan kuvailla seuraavasti:

- Luottamuksellisuus: Järjestelmässä oleva ja siihen liittyvä tieto suojataan luvattomalta käytöltä. Tyypillisesti tähän hyödynnetään esimerkiksi salausalgoritmejä.
- Eheys: Järjestelmässä oleva tieto ei saa muuttua oikeutettujen toimintojen välillä. Jos tieto muuttuu luvatta joko vahingossa tai hyökkäyksen vuoksi, muutos pitää havaita. Työkaluja tätä varten ovat esimerkiksi hajautusarvot.
- Saatavuus: Järjestelmässä oleva ja siihen liittyvä tieto, kuten myös sen muut resurssit ja toiminnallisuudet on oltava oikeutetun käyttäjän saatavilla niitä tarvittaessa (mahdollisimman pienellä viiveellä).

- Kiistämättömyys: Toimija ei voi menestyksekkäästi kiistää tekemäänsä. Työkaluja tätä varten ovat esimerkiksi logitiedostot ja tallentava kulun valvonta.
- Todentaminen: Toimijan identiteetti voidaan varmistaa.

Näiden elementtien saavuttamiseen ja määrittämiseen voidaan käyttää apuna esimerkiksi teollisuusstandardeja ja yrityksen tietoturvapolitiikkoja. Joskus kuitenkin elementtien yhteensovittaminen keskenään voi olla haasteellista. Esimerkiksi luottamuksellisuuden ja saatavuuden välille voi helposti muodostua ristiriita.

Vaikka digitaalista allekirjoitusjärjestelmää suunniteltaessa tulee kaikki nämä elementit huomioida, on myös hyvä ymmärtää, että digitaalisen allekirjoittamisen tarkoitus itsessään on todistaa siihen liitetyn tiedon eheys sekä todentaa tiedon lähde. Se siis toimii myös itse merkittävänä tietoturvaan liittyvänä tekijänä yrityksessä.

2.3.2. Allekirjoitusjärjestelmän potentiaaliset uhat

Koska digitaalinen allekirjoitusjärjestelmä toimii usein yhtenä omistajansa tietoturvaankkurina, se on potentiaalinen kohde erityyppisille ja eritasoisille hyökkäyksille. Uhka-analyysiin vaikuttaa suuresti järjestelmän toimintaympäristö, kuten minkälaisessa verkossa järjestelmä toimii sekä mitä laitteistoa ja ohjelmistoja järjestelmä käyttää.

Suojautuminen mietitään Intelillä aina ammattimaisia toimijoita vastaan. Toimijoilla oletetaan olevan merkittävästi ammattitaitoa, aikaa ja suuret taloudelliset resurssit. Tällaisilta toimijoilta suojauduttaessa suojaudutaan myös vähemmän ammattimaisilta rikollisilta, kuten aktivisteilta, pienrikollisilta ja harrastelijoilta. Yrityksen asianmukaisesti suojatussa sisäverkossa toimivan järjestelmän osalta suojauksen painopiste on kohdennetuissa hyökkäyksissä, joskaan opportunistisia massahyökkäyksiäkään ei voida kokonaan sivuuttaa. Julkisessa verkossa toimiva järjestelmä on alttiimpi myös massahyökkäyksille.

Alla on listattu joitakin potentiaalisia kohteita, jotka kiinnostavat allekirjoitusjärjestelmää vastaan toimivia hyökkääjiä. Kattavampi uhka-analyysi tulee tehdä jokaiselle järjestelmälle toteutuskohtaisesti.

- Yksityiset avaimet, jotka ovat allekirjoitusten luotettavuuden perusta.
- Käyttäjät ja heidän käyttäjätunnuksensa, joiden anastaminen voi mahdollistaa käyttäjän allekirjoitusoikeuksien hyödyntämisen.

- Käyttäjän ja allekirjoituspalvelimen välinen tietoliikenne, jota voidaan salakuunnella tai käyttää tiedon manipulointiin.
- Käyttöoikeusluettelot (engl. Access Control List, lyh. ACL), jotka määrittävät esimerkiksi valtuutettujen käyttäjien allekirjoitus- ja ylläpito-oikeudet.
- Seulontaan käytettävät skriptit, joita manipuloimalla voidaan kiertää tietoturvamäärityksiä.
- Allekirjoitettava tieto ja sen muuttaminen, mikä mahdollistaisi esimerkiksi haittakoodin allekirjoittamisen.
- Tapahtumakirjanpito eli lokit (engl. logs), joita muuttamalla hyökkääjä voi pyrkiä peittämään omia toimiaan.

Näitä kohteita vastaan hyökkääjällä on käytössään erinäisiä hyökkäystyyppejä. Brute Force -hyökkäyksien tavoite on murtaa järjestelmä löytämällä oikea syöte kaikkien mahdollisten syötteiden joukosta. Ideana on yksinkertaisesti yrittää syötteitä niin kauan, kunnes oikea osuu kohdalle. Allekirjoitusjärjestelmässä tällaisen hyökkäyksen kohteina ovat esimerkiksi allekirjoitusavaimet, kaikki salasanoilla suojattu materiaali (esim. käyttäjätunnukset) ja allekirjoitettavan tiedon hajautusarvot. Salasanoja vastaan voidaan käyttää Brute Force -periaatteella kehitettyjä astetta kehittyneempiä hyökkäysmalleja kuten sanakirjahyökkäyksiä. Hajautusarvojen murtamisessa pyritään nk. törmäyksiin (engl. hash collision), mikä perustuu siihen, että täysin saman hajautusarvon voi saada aikaiseksi muullakin kuin alkuperäisellä syötteellä. Tämä on vanha hyökkäystapa, jota esimerkiksi jo Alan Turing käytti purkaessaan Enigma-salauslaitetta toisessa maailmansodassa [12].

Mies välissä -hyökkäyksissä (engl. Man-in-the-Middle attack, lyh. MITM) hyökkääjä pyrkii kuuntelemaan ja mahdollisesti myös manipuloimaan tietoa kahden tietoliikenteen solmukohdan välillä. Allekirjoitusjärjestelmässä potentiaalinen kohde on asiakaslaitteen ja allekirjoituspalvelimen välinen yhteys. Myös SSL- ja TLS-salatut verkot ovat potentiaalisia kohteita näille hyökkäyksille niiden haavoittuvuuksien vuoksi, esimerkkinä Heartbleed [13] ja POODLE [14].

Palvelunestohyökkäys (engl. Denial-of-Service attack, lyh. DoS attack) pyrkii kuormittamaan palvelinta massiivisilla määrillä kutsuja, jotta kyseinen palvelin ei kykenisi

käsittelemään asiallisia kutsuja ja on täten saavuttamattomissa niille, jotka sitä tarvitsevat. Allekirjoitusjärjestelmän osalta tällaisen hyökkäyksen motiivi voisi olla esimerkiksi kilpailuhyödyn saavuttaminen häiritsemällä yrityksen tuotekehitys- ja tuotantoprosesseja.

Side Channel -hyökkäykset perustuvat tiedon keräämiseen kohdelaitteesta fyysisillä keinoilla. Periaatteessa tämä on mahdollinen hyökkäyskeino HSM:iä vastaan. Tällöin Side Channel -hyökkäys vaatii fyysisen pääsyn kohdelaitteeseen. [15]

2.3.3. Avainten suojaaminen

Yksityiset avaimet ovat allekirjoitusjärjestelmien turvallisuuden ja luotettavuuden perusta. Järjestelmän kannalta on elintärkeää säilyttää yksityinen avain ja tarvittaessa sen avainfraasi turvassa ulkopuolisilta. Mikäli avain on kokonaisuudessaan samassa paikassa, tulee avaimen haltijan olla luotettu. Usein etenkin yritysmaailmassa luottaminen yksittäisiin henkilöihin on kuitenkin riskialtista esimerkiksi erehtymisriskin ja henkilökohtaisten intressien vuoksi. Tämän takia on kehitetty protokollia jaettujen avainten luontiin (engl. Distributed key generation / Threshold cryptosystem). Nämä perustuvat ajatukseen, että yksittäisellä taholla ei ole pääsyä kokonaiseen salattuun avaimeen vaan ainoastaan yhteen sen osaan. Muut osat ovat muiden tahojen hallussa, ja avaimen käyttäminen vaatii näiden tahojen yhteistyötä salatun tiedon purkamiseen.

Avainten suojaaminen ilman, että sitä ikinä annetaan selkokielisenä kenenkään saataville, onnistuu HSM-laitteilla. Ne ovat laitteistokomponentteja, joiden koko vaihtelee USB-tikun ja laitetelineeseen (engl. rack) asennettavan kotelon välillä. HSM on järeä ja yleisesti luotettavana pidetty tapa salata yksityiset avaimet. Kuvassa 2.2 on Thalesin valmistama nShield Solo HSM. Kyseinen HSM kytketään tietokoneen PCI-express -väylään. Vastaavia on käytetty myös tässä toteutuksessa.

Kuva 2.2: Thales nShield Solo HSM

Yksityinen avain voidaan myös pyrkiä murtamaan ns. brute force -hyökkäyksellä ilman, että HSM ikinä pettää ja avaimen salaus murtuu. Hyökkääjä voi yrittää käydä läpi jokaisen luvun aina avaimen maksimikokoon asti, kunnes pystyy tuottamaan oikeanlaisen allekirjoituksen. Tältä suojaudutaan yksinkertaisesti valitsemalla tarpeeksi iso avaimen koko, koska koon kasvattaminen pitkittää eksponentiaalisesti purkamiseen vaadittavaa aikaa.

3. TOTEUTUS

Tämän luvun kohta 3.1. käy läpi toteutetulle järjestelmälle asetetut vaatimukset ja kohta 3.2. kuvailee vaatimusten pohjalta pohdittuja suunnitteluperiaatteita. Järjestelmään liittyvät roolit on määritetty kohdassa 3.3. Kohta 3.4. pyrkii antamaan kokonaiskuvan järjestelmän rakenteesta ja sen pääelementien toiminnasta periaatteellisella tasolla. Kohta 3.5. esittelee työnkulkua järjestelmän elementtien läpi ja kohta 3.6. selittää järjestelmän eri elementtien tarjoamia rajapintoja. Järjestelmän sisäistä tietomallia ja terminologiaa avataan kohdassa 3.7.. Tämän diplomityön kannalta merkittävä aihe on järjestelmän seulontaominaisuus, joka käsitellään omassa kappaleessaan 3.8. Kohta 3.9. käy läpi järjestelmän tietoturvan kannalta muita oleellisia asioita. Loppukäyttäjien hyödyntämien asiakasohjelmien toteutus ei varsinaisesti kuulu tämän diplomityön piiriin, mutta niiden esittely on tarpeellista järjestelmän kokonaiskuvan hahmottamiseksi. Kohta 3.10. käy lyhyesti läpi toteutettua järjestelmää hyödyntäviä ohjelmistotyökaluja.

3.1. Vaatimukset

Ensisijainen tarve toteutettavalle järjestelmälle oli saada mobiililaitteita kehittävät Intelin sisäiset tiimit käyttämään luotettua allekirjoitusjärjestelmää, jossa on keskitetty avainten-hallinta ja tuki kehitettävien sulautettujen järjestelmien komponenteille. Järjestelmä tuli suunnitella kuitenkin niin geneeriseksi, että se mahdollistaa Intelin muidenkin allekirjoitustarpeiden täyttämisen tulevaisuudessa mahdollisimman pienillä muutoksilla. Kiteytetty tavoite on parantaa Intelin, sen sopimusvalmistajien sekä loppukäyttäjien tietoturvaa.

Taulukkoon 3.1 on listattu yleiset vaatimukset, jota projektille asetettiin. Näiden vaatimusten lisäksi taulukkoon 3.2 on listattu asiakastiimien alkuvaiheessa asettamia teknisiä vaatimuksia, jotka ovat keskeisiä tiimien kehittämille tuotteille. Tiedostomuodot ja niiden otsakkeet ovat Intelin omia ja niiden nimet on muutettu yrityssalaisuuden vuoksi. Näihin tiedostomuotoihin viitataan tässä kohdassa nimillä A, B ja C.

Taulukko 3.1: Yleisiä vaatimuksia

Kategoria	Vaatimus
Allekirjoitusavaimet	Järjestelmän on tuettava RSA-avaimia minimikooltaan kaksi kilobittiä
Allekirjoitusavaimet	Allekirjoitusavaimet on suojattava vahvalla salauksella koko niiden elinkaaren ajan
Allekirjoitusavaimet	Allekirjoitusavainten hallinta ja niiden käyttäminen allekirjoittamiseen on oltava keskitetty allekirjoituspalvelimille
Allekirjoituspalvelimet	Allekirjoituspalvelimilla on oltava korkea (>99%) saatavuusaste (engl. high availability)
Allekirjoituspalvelimet	Allekirjoituspalvelinten on pystyttävä allekirjoittamaan binäärimuotoista tietoa tiedostomuodoista piittaamatta
Allekirjoituspalvelimet	Allekirjoituspalvelinten on pystyttävä validoimaan allekirjoituspyyntö ja siihen liittyvän allekirjoitettavan tiedon oikeellisuus ennen allekirjoittamista
Allekirjoituspalvelimet	Allekirjoituspalvelinten on palautettava vastauksessaan allekirjoituksen lisäksi myös käytetyn avainparin julkinen avain
Asiakasohjelmisto	Käyttäjien pitää pystyä lähettämään allekirjoituspyyntöjä Ubuntu 12, Ubuntu 14, Windows 7, Windows 8, Windows 8.1 ja Windows 10 -ympäristöistä
Asiakasohjelmisto	Asiakasohjelmiston on toimittava myös automaattisilla käännöspalvelimilla ilman ihmisten jatkuvaa vuorovaikutusta
Asiakasohjelmisto	Asiakasohjelmiston on kyettävä todentamaan allekirjoitus vastaanottamallaan julkisella avaimella
Infrastruktuuri	Järjestelmään liitettyjen palvelimien on kirjoitettava auditointilogeja kaikista ihmisten suorittamista toimenpiteistä (sis. ylläpito- ja kulutustoimenpiteet)
Infrastruktuuri	Järjestelmä toimii Intelin omassa sisäisessä verkossa TLS-suojattujen yhteyksien läpi käyttäen Intelin oman PKI-palvelun jakamia varmenteita
Infrastruktuuri	Järjestelmään liitettyjen palvelimien on täytettävä Intelin asettamat fyysiset suojavaatimukset
Infrastruktuuri	Järjestelmään on pystyttävä lisäämään kapasiteettia käyttöasteen noustessa
Ylläpito	Vain oikeutetut ylläpitäjät voivat tehdä muutoksia järjestelmään
Ylläpito	Tarve luottaa yksittäiseen ylläpitäjään on minimoitava

Alkuvaiheessa vaatimuksien kerääminen vaati paljon viestintää tuotekehitystiimien sekä tietoturvavastaavien kanssa. Yhteenvetona teknisistä vaatimuksista voidaan todeta, että ne olivat eri tuotekehitystiimeillä varsin yhtenäiset. Allekirjoittamiseen käytetään kahta eri algoritmia (PKCS#1 v1.5 sekä PKCS#1 v2.1 PSS) ja hajautusarvojen laskemiseen ainoastaan SHA256-algoritmia.

Taulukko 3.2: Asiakastiimien erityisvaatimuksia

Tuote / tuoteperhe	**Vaatimus**
Merrifield ja Moorefield	Allekirjoittamiseen käytettävä RSA:n PKCS#1 v1.5-algoritmia
Merrifield ja Moorefield	Hajautusarvojen laskentaan käytettävä SHA256-algoritmia
Merrifield ja Moorefield	Järjestelmän on tuettava tiimien omaa tiedostomuotoa (otsake A)
Merrifield ja Moorefield	Allekirjoitus on laskettava tiedoston otsakkeen sekä tiedoston kuorman hajautusarvon ylitse
Merrifield ja Moorefield	Tietomallin ja pääsynvalvonnan on tuettava sitä, että yksittäisen projektin alla voi olla useita allekirjoitettavia komponentteja sekä allekirjoitusavaimia
Merrifield ja Moorefield	Järjestelmän tieto- ja tietoturvamallin on tuettava sitä, että kaksi tai useampi eri allekirjoittaja voi käyttää samaa allekirjoitusavainta eri komponenttien allekirjoittamiseen
Modeemit ja antennit	Allekirjoittamiseen käytettävä RSA:n PKCS#1 v1.5-algoritmia
Modeemit ja antennit	Hajautusarvojen laskentaan käytettävä SHA256-algoritmia
Modeemit ja antennit	Järjestelmän on tuettava tiimien omaa tiedostomuotoa (otsake B)
Broxton	Allekirjoittamiseen käytettävä RSA:n PKCS#1 v2.1 PSS-algoritmia
Broxton	Hajautusarvojen laskentaan käytettävä SHA256-algoritmia
Broxton	Järjestelmän on tuettava tiimien omaa tiedostomuotoa (otsake C)

3.1.1. Mitä allekirjoitetaan?

Alkutilanteessa merkittävimmät asiakkaat järjestelmälle olivat Merrifield ja Moorefield -mobiilialustoja kehittävät tiimit. Molemmat tuotteet hyödyntävät samantyyppistä varusohjelmatason tietoturva-arkkitehtuuria, joka perustuu omaan suorittimeensa laitteella. Tätä tietoturvasuoritinta konfiguroidaan konfiguraatiopaketeilla ja kehitetään varusohjelmakoodilla. Kaikki nämä ovat omia komponenttejaan, jotka tulee allekirjoittaa. Kaikki kyseiset komponentit sisältävät A-tiedosto-otsakkeen (yksinkertaistettu kuva 3.1), joka sisältää

metatietoa allekirjoitettavasta komponentista, sen allekirjoittajasta sekä vastaanottajasta. Otsaketta seuraa aina varsinainen kuorma (engl. payload), joka voi olla esimerkiksi varusohjelma tai konfiguraatiopaketti (engl. token). [16]

Kuten kuvasta 3.1 nähdään, A-otsake sisältää jo itsessään hajautusarvon kuormasta. Varsinainen allekirjoitus lasketaan tämän hajautusarvon sekä itse otsakkeen yli. Todentaja (tässä tapauksessa laitteen tietoturvasuoritin) voi todentaa allekirjoituksen hallussaan olevalla julkisella avaimella ja verrata tulosta itse laskemiinsa hajautusarvoihin.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Allekirjoituksen aikaleima																															
Allekirjoittajan tunniste																															
Allekirjoitusavaimen tunniste																															
Sopimusvalmistajan tunniste																															
Komponentin tunniste																															
Suora muistiosoite allekirjoitukseen																															
Hajautusarvoalgoritmi										Julkisen avaimen tyyppi						varattu															
Julkinen RSA-avain (256 tavua)																															
Barret n -arvo (20 tavua)																															
Komponentin versio																															
Viimeisen tietoturvapäivityksen versio																															
Kuorman hajautusarvo (16 tavua)																															
Kuorman latausosoite																															
Kuorman koko																															
Allekirjoitus (256 tavua)																															

Kuva 3.1: A-tiedosto-otsakkeen yksinkertaistettu rakenne

Otsakerakenteessa on mukana oma 256 tavun kenttä julkiselle avaimelle, johon käytetyn allekirjoitusavainparin julkinen avain laitetaan. Laite laskee tästä julkisesta avaimesta hajautusarvon ja vertaa sitä laitteelle aikaisemmin provisioituun arvoon. Luottamusketju (engl. trust chain) Moorefield- ja Merrifield-laitteille asetetaan niin, että tärkeimpien julkisten avainten hajautusarvot poltetaan tuotantovaiheessa laitteella oleville sulakkeille. Tällöin niitä ei voida enää ohjelmallisesti muuttaa. Sulakkeet ovat kalliita, ja sen vuoksi näin tehdään vain muutamalle tärkeimmälle avaimelle. Tärkeimpinä avaimina pidetään avaimia, joilla allekirjoitetut konfiguraatiopaketit asettavat laitteelle kriittisiä tietoturva-

asetuksia, kuten:

- kloonauksen esto
- version palautuksen esto
- uudelleen konfiguroitavien (eli tietoturvasuorittimen muistiin tallennettavien) julkisten avaimien asettaminen
- hyväksyttävät minimiversiot eri elementeistä
- vahtikoirien asetukset.

Vähemmän kriittisiä konfiguraatiopaketteja, jotka allekirjoitetaan uudelleen ohjelmoitavilla avaimilla, asettavat esimerkiksi laitteen modeemien MAC-osoitteet, sarjanumerot ja DRM-avaimet. Näitä vähemmän tietoturvaherkkiä konfiguraatiopaketteja voidaan allekirjoittaa useammilla eri avaimilla, esimerkiksi Intelin omilla ja sopimusvalmistajien avaimilla. Tämä on hyvä esimerkki tilanteesta, jossa insinöörien allekirjoitusoikeuksia ei voida rajata ainoastaan yksittäiseen avaimeen tai komponenttiin. Näiden allekirjoitusten avainten julkisen parin hajautusarvot provisioidaan laitteelle konfiguraatiopaketeilla, jotka allekirjoitetaan sulakkeille poltettujen allekirjoitusavainten yksityisellä parilla.

A-rakenteen lisäksi muita allekirjoitettavia tiedostorakenteita ovat modeemi- ja antennitiimien käyttämä B sekä Broxton-alustassa käytetty C. Modeemi- ja antennituotteet ovat huomattavasti yksinkertaisempia tietoturva-arkkitehtuuriltaan, ja niissä projekteissa allekirjoitetaan aina vain yhtä komponenttia yhdellä avaimella [17]. Ainoa vaatimus on, että tuotekehitys- ja tuotantoavaimet pitää pystyä luomaan erikseen. Broxton-mobiilialusta käyttää Merrifieldin ja Moorefieldin tavoin konfiguraatiopaketteja, mutta sen tiedostootsake on yksinkertaisempi [18].

3.1.2. Toimintaympäristön rajoitteet

Kryptografiset toimenpiteet vaativat suhteellisen paljon laskentatehoa, muistia tai molempia. Sulautettujen ja mobiilijärjestelmien tapauksessa liikutaan kuitenkin ympäristössä, jossa tyypillisesti resurssit ovat erittäin rajalliset työpöytä- ja palvelinympäristöön verrattuna. Tämä asettaa tuotekehitykselle arkkitehtuurisia haasteita, sillä tietoturvan osalta ei voida tehdä kompromisseja linjattujen vähimmäisvaatimusten suhteen.

Allekirjoitusjärjestelmää suunniteltaessa on ymmärrettävä nämä rajoitteet. Asiakastiimit ovat pyrkineet ottamaan rajoitteet huomioon vaatimuksia koottaessa.

3.2. Suunnitteluperiaatteet

Toteutuksessa on pyritty täyttämään kaikki asetetut vaatimukset niin, että järjestelmä on samaan aikaan sekä tietoturvallinen että tuotantokelpoinen. Toteutettavan allekirjoitusjärjestelmän on tarkoitus parantaa Intelin ja sen asiakkaiden valmistamien mobiililaitteiden kuten älypuhelimien, tablet-tietokoneiden ja puettavien laitteiden tietoturvaa. Järjestelmällä allekirjoitettava tieto on pääsääntöisesti sulautettuja ohjelmistokomponentteja, käyttöjärjestelmäkoodia ja konfiguraatiopaketteja.

Prosessin kannalta monimutkaisuutta lisää se, että allekirjoittavia tahoja samalle laitteelle on monia. Intelin omat tuotekehittäjät allekirjoittavat komponentteja tuotekehitysmielessä ja tarvitsevat rajoitusten kannalta enemmän joustavuutta kuin esimerkiksi asiakasrajapinnassa toimivat työntekijät, jotka voivat allekirjoittaa komponentteja suoraan sopimusvalmistajille tuotantokäyttöön. Jotkut komponentit sopimusvalmistajat allekirjoittavat itse.

3.2.1. Skaalautuvuus

Pitkän käyttöiän ja tulevaisuudessa muuttuvien vaatimusten vuoksi järjestelmän haluttiin skaalautuvan tarvittavan kapasiteetin mukaan. Tarkoituksena oli, että asiakasmäärä voi kasvaa kymmeniin tuhansiin ilman merkittäviä muutoksia sovelluksiin. Myös projektien lukumäärä ja niiden alle hierarkisesti konfiguroitavien komponenttien, avainten ja näiden sidosten määrien täytyi olla joustavia. Jotkut tuotteet ja projektit sisältävät vain yhden allekirjoitettavan komponentin, kun taas toiset voivat sisältää useita kymmeniä. Samoin avaimia voi lähtötietojen mukaan olla 1-20 projektia kohden.

Järjestelmän oli myös tuettava erilaisia tiedostotyyppejä eli tiedostoja eri otsakkeilla. Uusien tiedostotyyppien lisääminen oli tehtävä mahdollisimman vaivattomaksi. Eri otsaketyypit vaihtelevat rakenteeltaan. Osa on staattisia eli kiinteäkokoisia tiedostoja, kun taas toisissa on monimutkaisempia sisäisiä ryhmittelyitä ja vaihtelevia määriä alaotsakkeita. Tämän vuoksi todettiin perustelluksi siirtää vastuu asiakasohjelmien toteuttamisesta erillisille työkalutiimeille. Heille kuitenkin tarjotaan yleispätevä ja dynaamisesti linkittyvä

asiakaskirjasto mahdollistamaan toimenpiteet allekirjoitusjärjestelmässä.

3.2.2. Avaintenkäyttöpolitiikka

Koska Intelin tuotteena syntyvälle laitteelle tallennettavien avainten määrä on rajallinen ja allekirjoitettavien komponentteja määrä on usein suuri, ei kaikkia allekirjoitettavia komponentteja voida allekirjoittaa eri avaimilla. Tästä syystä Intelin tuotekehitystiimien tietoturva-arkkitehdit ovat määrittäneet avaintenkäyttöpolitiikan (engl. key usage policy), joka määrittää, mikä komponentti allekirjoitetaan milläkin avaimella.

Kuva 3.2 esittää kuvitteellisen projektin tai tuotteen avaintenkäyttöpolitiikkaa, joka on kuvattu matriisina. Todellisuudessa projekteissa voi olla useita kymmeniä allekirjoitettavia komponentteja ja tyypillisesti 1-20 käytettävää avainta, joten tosielämän matriisi voi täten olla merkittävästi havainnekuvaa suurempi.

	Kiinteät		Konfiguroitavat (Intel)		Konfiguroitavat (asiakas)	
	Avain 1	**Avain 2**	**Avain 3**	**Avain 4**	**Avain 5**	**Avain 6**
Komponentti 1	x					
Komponentti 2	x					
Komponentti 3		x				
Komponentti 4			x		x	
Komponentti 5			x	x	x	x

Kuva 3.2: Havainnekuva esimerkkituotteen avaimenkäyttöpolitiikasta

Avaimet on kategorioitu sen perusteella, miten julkinen avain tuotteeseen asetetaan. Kiinteät avaimet asetetaan valmistusvaiheessa niin, että niitä ei voida käytännössä myöhemmin muuttaa. Konfiguroitavat julkiset avaimet asetetaan ohjelmallisesti tuotteen muistiin, joko itse Intelin tai sen sopimusvalmistajan (kuvassa "asiakas") toimesta. Julkisten avainten asettamiseen laitteelle tapahtuu konfiguraatiopaketeilla.

Vaikka samalla avaimella pitää pystyä kirjoittamaan useita eri komponentteja, ei tämä välttämättä ole sallittua yksittäisen henkilön toimesta, vaan oikeudet haluttiin hajauttaa. Tämä ratkaistiin tietomallilla, joka ei sido allekirjoitusoikeuksia yksittäiseen avaimeen tai komponenttiin vaan näiden yhdistelmään.

3.2.3. Seulonnan periaatteet ja tavoitteet

Seulonta (engl. screening) on palvelinpuolen ominaisuus, joka haluttiin lisätä parantamaan tietoturvaa. Seulonnan toimintatarkoitus on lukea allekirjoitettava tieto ja tarkistaa määrättyjen ehtojen täyttyminen ennen itse allekirjoitusproessia. Jos seulonta ei mene läpi, eli määrätyt ehdot eivät täyty, allekirjoitusta ei tehdä ja käyttäjälle annetaan virheilmoitus. Olennaisia kohteita seulonnalle ovat tiedostojen otsakkeet, jotka sisältävät metatietoa allekirjoitettavasta komponentista. Tämä metatieto yleensä määrittää laitteen suhtautumisen tietopakettiin, jota ladataan laitteeseen. Joissakin tapauksissa, kuten konfiguraatiopakettien kohdalla, otsakkeiden lisäksi myös itse tietokuormaa halutaan seuloa.

Koska useat eri tahot (Intelin omat eri tiimit asiakasrajapinnasta tuotekehitykseen, alihankkijat ja asiakkaat) allekirjoittavat samasta tuotteesta eri osia osittain samoilla avaimilla, halutaan seulonnalla estää sekä tahallisia että tahattomia väärinkäytöksiä. Karkeasti seulonnalla on kolme käyttötarkoitusta: estää hyökkäyksiä, ehkäistä inhimillisiä virheitä ja asettaa automatisoidusti arvoja tiedostoon.

Seulontaskripteillä voidaan asettaa automaattisesti esimerkiksi arvoja, jotka mahdollistavat allekirjoituksen jäljitettävyyden aikaleiman ja eri tunnisteiden avulla. Näin ollen allekirjoitustapahtuma on yksilöitävissä, jos siihen on tarvetta.

Ajatus on, että toteutettava järjestelmä tarjoaa toiminnallisuudet, jotka mahdollistavat projektien omistajien omien seulontasääntöjen määrittämisen ja luonnin korvaamaan vastuuhenkilöiden käsin tekemiä tarkastuksia allekirjoituspyyntöjä hyväksyessään. Tämä nopeuttaa merkittävästi tuotekehitys- ja asiakastukiprosesseja, kun sekä sisällön validointi että hyväksyminen voidaan suorittaa automaattisesti.

Koska seulontatoiminnallisuus toimii itsessään ikään kuin portinvartijana allekirjoitusjärjestelmän läpi kulkevalle tiedolle, on se merkittävä potentiaalinen kohde hyökkäyksille. Seulontajärjestelmän asianmukainen suojaaminen on siis tärkeää.

3.2.4. Tunnistettuja erityishaasteita

Skaalautuvuuden tarpeen ennustaminen oli hankalaa. HSM:t ja tuotantokelpoiset palvelimet ovat kalliita. Laitehankintoja tehtäessä tarvittavan lukumäärän arvioiminen ei ollut helppoa.

Seulontatoiminnallisuus aiheutti puolestaan pohdittavaa verkon kuormittumisen suh-

teen. Tiedon seulonta edellyttää, että seulottava tieto lähetetään kokonaisuudessaan palvelimelle eikä pelkkä hajautusarvo riitä. Ratkaisu tähän ongelmaan oli määritellä kaksi vaihtoehtoista työnkulkua allekirjoituspyynnöille.

Lisäksi tietoturvan kannalta pidettiin järkevänä, että järjestelmän rakenne pyritään pitämään yksinkertaisena. Monimutkaisuuden lisääntyessä järjestelmän on silti oltava luotettu. Fyysisen turvallisuuden osalta haastetta aiheuttivat tehtaiden tuotantolinjat, joita ei voitu pitää turvallisena paikkana säilöä allekirjoittamiseen liittyvää salaista materiaalia kuten käyttäjätunnuksia, allekirjoitusvaimia tai etenkään HSM-laitetta.

3.3. Roolit

Järjestelmän hallinta- ja käyttöoikeudet on pyritty jakamaan eri rooleihin niin, että yksittäinen henkilö ei voi päätyä tahallisesti tai vahingossa järjestelmän yksittäiseksi koko toiminnan vaarantavaksi uhaksi (engl. single-point-of failure, lyh. SPOF). Tällä vastuunjaolla (engl. separation of duties, lyh. SoD) pyritään siis estämään sekä inhimillisiä virheitä että sosiaalista manipulointia (engl. social engineering).

Järjestelmään liittyvät roolit on listattu taulukossa 3.3. Järjestelmän ohjelmakoodi tunnistaa näistä ainoastaan neljä roolia: sovellusylläpitäjän, SPID-isännän ja allekirjoittajan. Allekirjoittaja-roolista järjestelmä osaa erottaa toisistaan luonnollisen henkilön ja käännöspalvelimen. Muut roolit ovat määritelty Intelin muissa järjestelmissä ja prosesseissa.

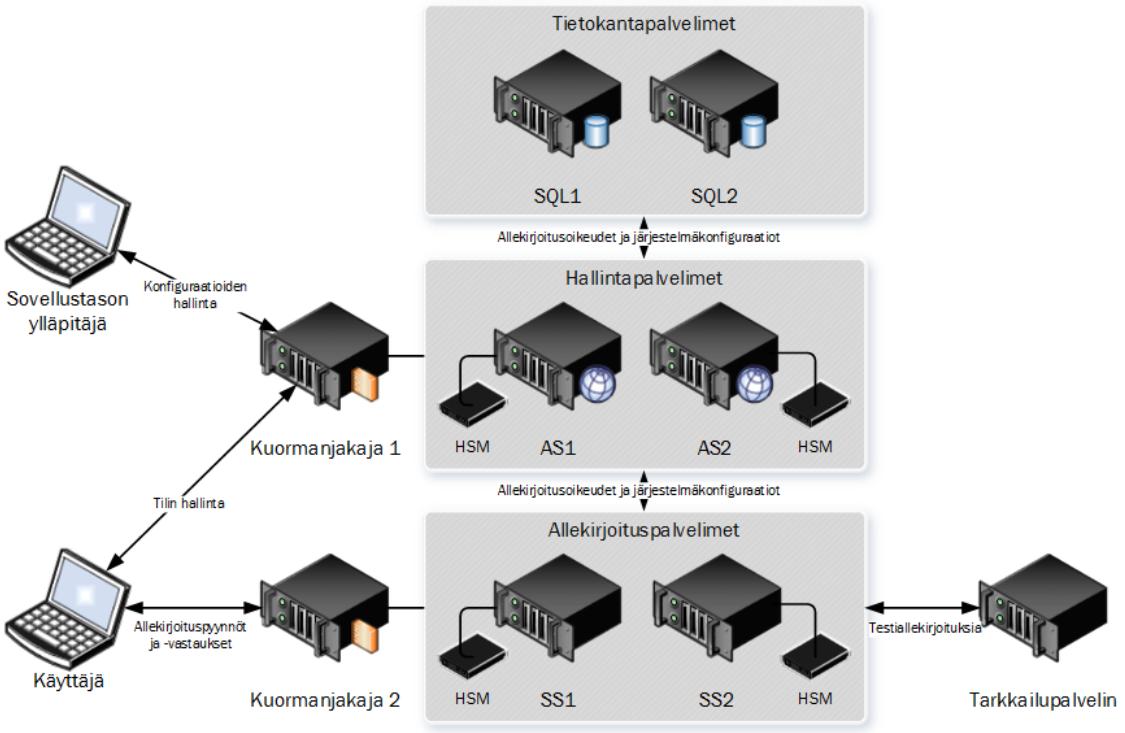
3.4. Korkean tason järjestelmäkuvaus

Infrastruktuuri on rakennettu Intelin omaan sisäverkkoon. Palvelimet keskustelevat keskenään web-rajapintojen ja -palveluiden kautta. Kaikki yhteydet ovat TLS-suojattuja. Kuva 3.3 esittää infrastuktuurin rakennetta korkealla tasolla. Siitä nähdään myös infrastruktuuriin kuuluvat palvelintyypit.

Järjestelmän verkkotopologiasta sekä palvelinten ja verkon fyysisestä hallinnoinnista vastaa Intelin IT-osasto. Myös valitut järjestelmäylläpitäjät ovat IT-henkilökuntaa. Palvelimet on sijoiteltu maantieteellisesti eri maihin niin, että niiden käyttöviiveet saadaan mahdollisimman lyhyiksi tärkeimmille tuotekehitystiimeille.

Taulukko 3.3: Roolit

Rooli	Kuvaus
Sovellusylläpitäjä (engl. application admin)	Henkilö, joka hallinnoi palvelimia sovellustasolla. Sovellusylläpitäjä pystyy luomaan projekteja, avaimia, seulontaskriptejä ja allekirjoituskonfigraatioita. Hän pystyy myös myöntämään allekirjoitusoikeuksia muille paitsi itselleen.
SPID-isäntä (engl. SPID manager)	Henkilö, joka hallinnoi oman liiketoimintayksikkönsä allekirjoitusprojektien konfiguraatioita. Siinä missä sovellusylläpitäjä toimii koko järjestelmän alueella, SPID-isäntä hallinnoi vain tiettyjä projekteja. Hän ei myöskään voi luoda uusia avaimia. Tämä rooli on tehty helpottamaan sovellusylläpitäjien työkuormaa.
Allekirjoittaja (engl. signer)	Henkilö tai henkilön hallinnoima käännöspalvelin, joka käyttää järjestelmää allekirjoitusten tuottamiseen. Yleisesti kuka tahansa yrityksen oma työntekijä pystyy pyytämään itselleen joitakin allekirjoitusoikeuksia. Allekirjoitusoikeudet saanut allekirjoittaja pystyy lähettämään järjestelmään allekirjoituspyyntöjä allekirjoitustyökalulla.
Järjestelmäylläpitäjä (engl. system admin)	Henkilö, joka hoitaa palvelimien laitteistoon, käyttöjärjestelmään ja verkkoon liittyvät asennukset.
Todentaja (engl. verifier)	Henkilö, joka automatisoi ja konfiguroi allekirjoituksen ohjelmallisen todentamisen lopputuotteessa. Todentaja tarvitsee allekirjoitukseen käytetyn avainparin julkisen avaimen.
Ylläpitokortin haltija (engl. admin card holder)	Henkilö, joka pitää hallussaan yhtä HSM:n ylläpitokorttia ja toisen kortin salasanaa.
Tietoturvavastaava (engl. security champion)	Henkilö, joka määrittää käytettävät tietoturvapolitiikat omalla vastuualueellaan. Järjestelmään vaikuttavia tietoturvavastaavia voi olla useita. Esimerkiksi liiketoimintayksikön (engl. business unit) tietoturvavastaava määrittää allekirjoitusprojektin (SPID) tietoturvakäytännöt eli esimerkiksi kriteerit, miten allekirjoitusoikeuksia jaetaan.
Tietoturva-auditoija (engl. security auditor)	Henkilö, joka suorittaa määräaikaisia tarkastuksia järjestelmän nykytilasta. Hänen vastuullaan on varmistaa, että järjestelmän kaikki osapuolet ovat hoitaneet tehtävänsä moitteetta ja järjestelmä läpäisee edelleen sille määrätyt tietoturvaan liittyvät kriteerit.



Kuva 3.3: Infrastruktuuri

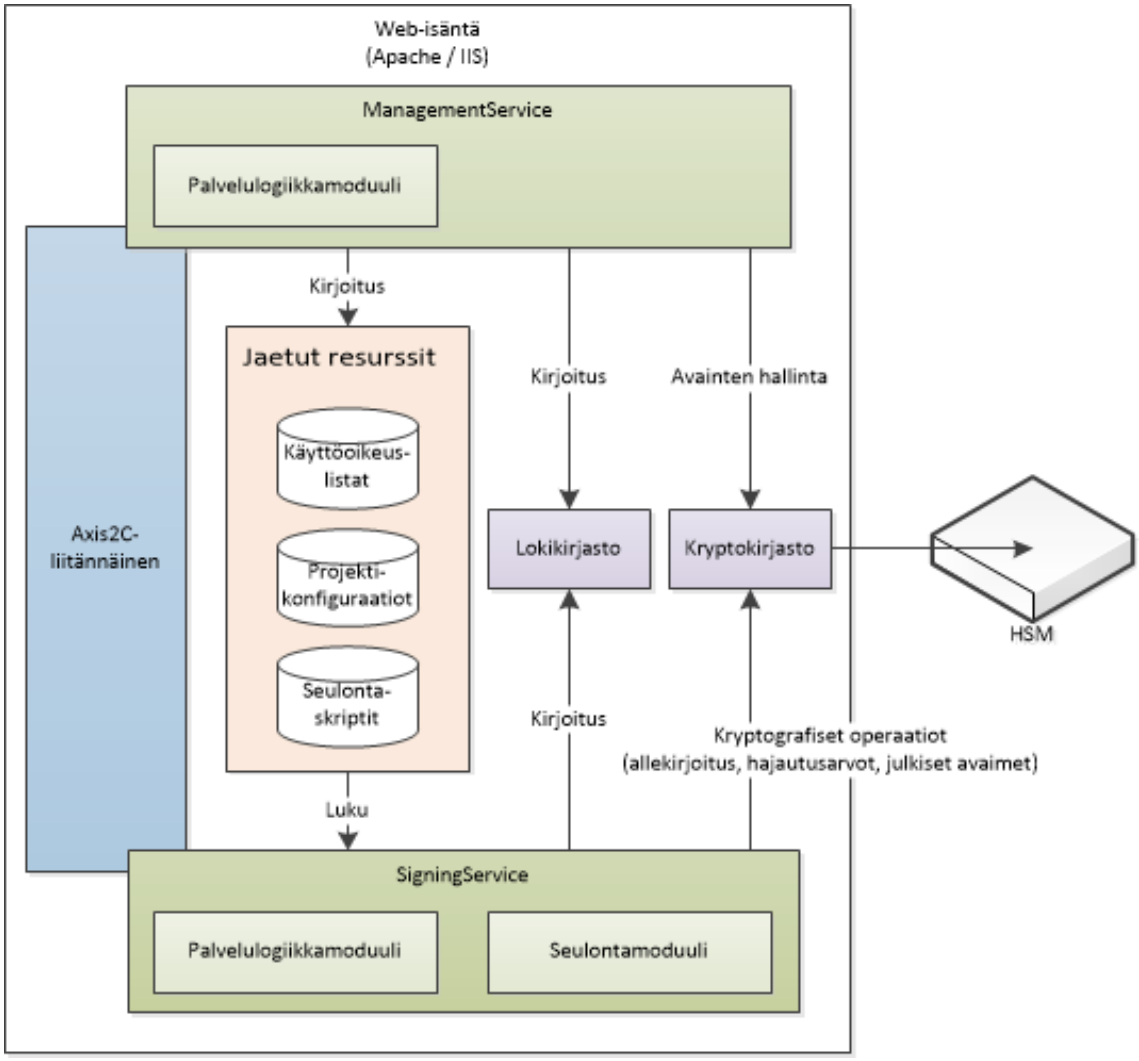
3.4.1. Allekirjoituspalvelimet

Allekirjoituspalvelimet suorittavat seulonnan ja allekirjoitusoperaatiot. Ohjelmistojen asennus tapahtuu paikallisesti, mutta asennuksen jälkeinen hallinta, kuten avainten lisääminen, tapahtuu hallintapalvelimien kautta. Allekirjoituspalvelimen ohjelmiston loogisia elementtejä on havainnollistettu kuvassa 3.4.

Allekirjoituspalvelimet pystyvät toimimaan itsenäisesti, vaikka yhteyttä hallinta- ja tietokantapalvelimiin ei olisikaan. Käyttöoikeudet on tallennettu paikallisesti, ja myös HSM:n avulla salatut allekirjoitusavaimet ovat paikallisesti tallennettuna tiedostojärjestelmässä.

Jokainen allekirjoituspalvelin tarjoaa kaksi web-palvelua: SigningService ja ManagementService. Ensimmäinen kuuntelee asiakasohjelmilta tulevia allekirjoituspyyntöjä ja jälkimmäinen käsittelee hallintapalvelimilta tulevia konfiguraatiopaketteja. Allekirjoituspalvelinohjelmistosta on versiot Windowsille sekä Linuxille (Red Hat ja Debian). Windows-versio hyödyntää Microsoftin IIS-palvelinohjelmistoa ja Linux-versiot Apache-palvelinohjelmistoa.

Molemmat web-palvelut käyttävät avoimen lähdekoodin Axis2C-



Kuva 3.4: Allekirjoituspalvelimen havainnekuva

ohjelmistokomponenttia. Tämä komponentti tarjoaa joukon C-kielisiä kirjastoja, joilla voidaan toteuttaa SOAP-tietoliikenneprotokollaan pohjautuvia web-rajapintoja. Sen pääkehittäjä on Apache Software Foundation. [19]

3.4.2. Hallintapalvelimet

Hallintapalvelimet tarjoavat web-käyttöliittymän sekä sovellustason ylläpitäjille että allekirjoittajille. Ylläpitäjät voivat muun muassa luoda ja muuttaa erilaisia konfiguraatioita, luoda avaimia ja lisätä allekirjoituspalvelimia keskitetyn hallinnan alle. Allekirjoittajat voivat luoda itselleen tilin ja pyytää allekirjoitusoikeuksia.

Hallintapalvelimia on kaksi ja ne ovat keskenään identtisiä. Ne käyttävät tietovarastoinaan tietokantapalvelimia, jotka on myös kahdennettu. Hallintapalvelimissa on kussakin oma HSM, joka toimii lähdesäiliönä allekirjoituspalvelimille lähetettäville allekirjoitusa-

vaimille. Allekirjoituspalvelimissa tallennettuna olevat avaimet ovat aina hallintapalvelimissa olevien avainten alijoukko. Ylläpitäjät voivat päättää, mitkä avaimet millekin allekirjoituspalvelimelle lähetetään.

Hallintapalvelimet säilövät sovellustason ylläpitäjien tekemät konfiguraatiot tietokantapalvelimille, jotka toimivat lähteenä json-muotoisille allekirjoituspalvelimille lähetettäville konfiguraatiopaketeille.

3.4.3. Kuormanjakajat

Kuormanjakajat ovat osa Intelin olemassa olevaa verkkoinfrastruktuuria. Ne ovat välityspalvelimia, jotka ohjaavat järjestelmään tulevat pyynnöt eteenpäin vähiten kuormitetulle tai nopeimmin vastaavalle palvelimelle. Kuormanjakajien tärkeä tehtävä on myös tarkistaa, että kohdepalvelin vastaa pyyntöihin ja estää pyynnön lähettämisen vialliselle palvelimelle.

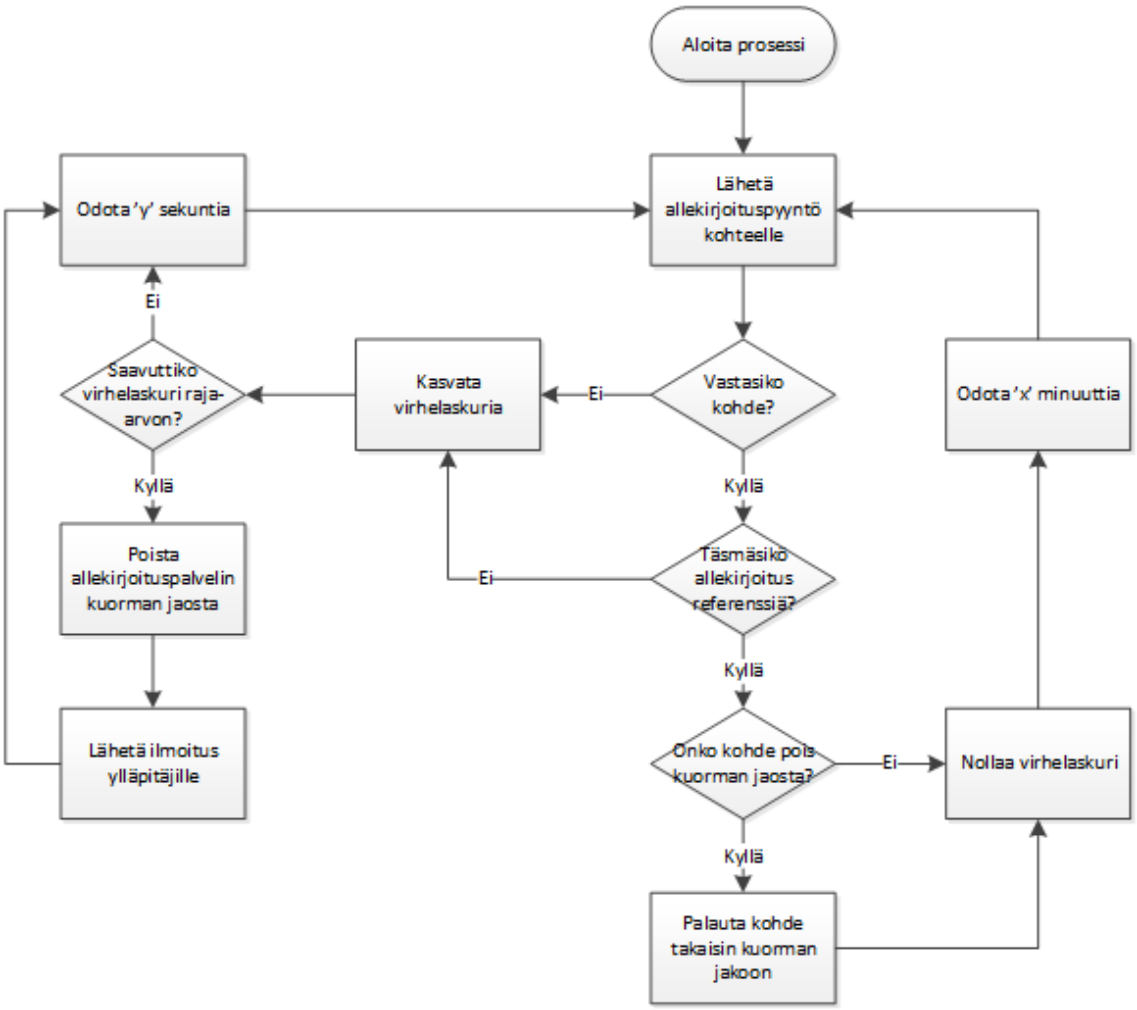
Allekirjoitusjärjestelmässä on omat kuormanjakajansa hallinta- ja allekirjoituspalvelimille. Koska allekirjoituspalvelimet ovat maantieteellisesti hajautettuja, kuormanjakaja ohjaa kutsun nopeimmin vastaavalle palvelimelle. Molemmat hallintapalvelimet puolestaan sijaitsevat samassa datakeskuksessa, joten niiden kohdalla kuormanjakaja lähettää kutsun vähemmän kuormitetulle palvelimelle.

Kuormanjakajat ovat ohjelmoitavissa lähettämään erilaisia kutsuja kohdepalvelimille. Yksinkertaisimmillaan kuormanjakaja lähettää palvelinryhmälle ping-pyynnön ja ohjaa pääkutsun nopeimmin pyyntöön vastanneelle palvelimelle. Tässä toteutuksessa kuormanjakajat on ohjelmoitu lähettämään HTTP-pyyntöjä (engl. HTTP request) ja tulkitsemaan kohdepalvelimilta palaavat HTTP-vastaukset (engl. HTTP response).

3.4.4. Tarkkailupalvelin

Intelin käyttämän kuormanjakajaohjelmiston rajallisuuden vuoksi se ei osaa löytää sovellustason virheitä. Tarkkailupalvelimen tehtävä on valvoa allekirjoituspalvelimien tuottamien allekirjoitusten oikeellisuutta. Tarkkailupalvelin lisättiin myöhemmässä vaiheessa sen jälkeen, kun yhden allekirjoituspalvelimen huomattiin lähettävän joskus virheellisiä allekirjoituksia. Näissä tapauksissa kuormanjakajat eivät huomanneet virheellistä toimintaa, vaan jatkoivat pyyntöjen ohjaamista virheellisesti toimivalle palvelimelle.

Tarkkailupalvelimen toimintaperiaate on yksinkertainen. Se lähettää säännöllisin väliajoin suoria allekirjoituskutsuja allekirjoituspalvelimille, minkä jälkeen se vertaa tulosta referenssidataan. Jos tulos ei täsmää referenssidataan, tarkkailupalvelin ilmoittaa kuormanjakajalle virheellisen allekirjoituspalvelimen sekä lähettää sähköposti-ilmoituksen järjestelmän ylläpidolle. Tämä logiikka on kuvattu kuvan 3.5 vuokaaviossa.

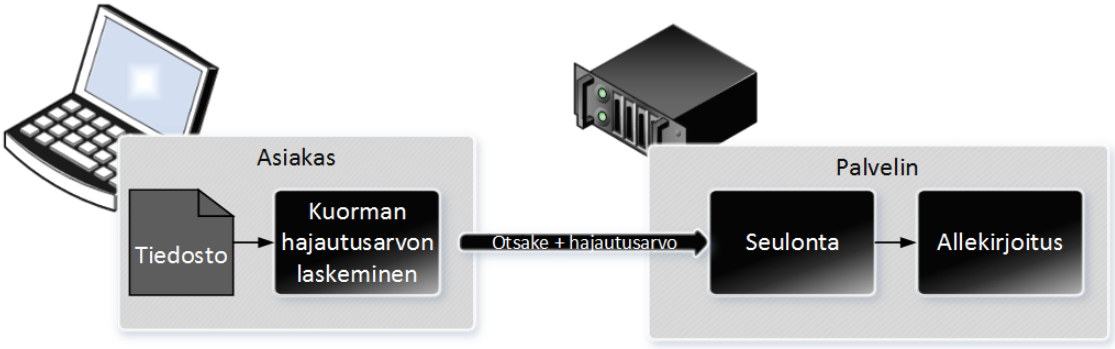


Kuva 3.5: Tarkkailupalvelimen toiminnan vuokaavio

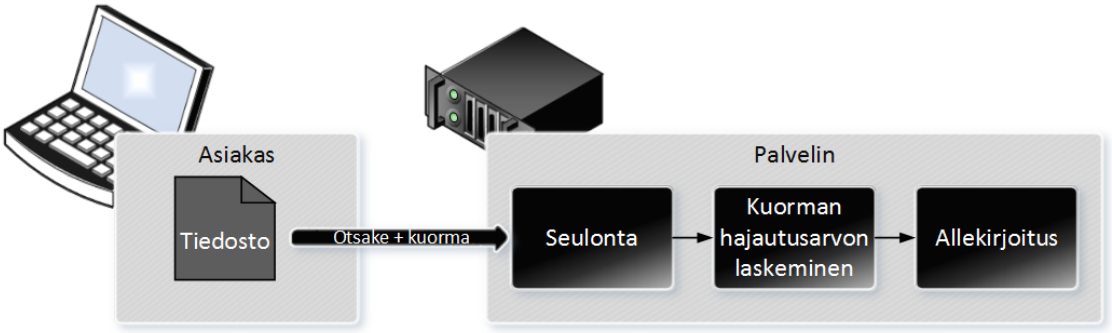
3.5. Työnkulku

Koska seulonta vaatii seulottavan tiedon lähettämistä allekirjoituspalvelimelle, verkon kuormitus kasvaa. Tämän vuoksi haluttiin tehdä kaksi eri työnkulkumahdollisuutta allekirjoituspyynnölle. Niissä tapauksissa, joissa halutaan seuloa pelkkää tiedoston otsaketta, allekirjoituspalvelimille lähetetään pelkästään otsake sekä kuorman hajautusarvo (kuva 3.6). Tämä pätee myös silloin, kun ei haluta seuloa lainkaan. Jos kuormaa halutaan

seuloa (esimerkiksi tietyt konfiguraatiopaketit), allekirjoituspalvelimelle lähetetään koko tiedosto (kuva 3.7).



Kuva 3.6: Pelkän otsakkeen seulonta



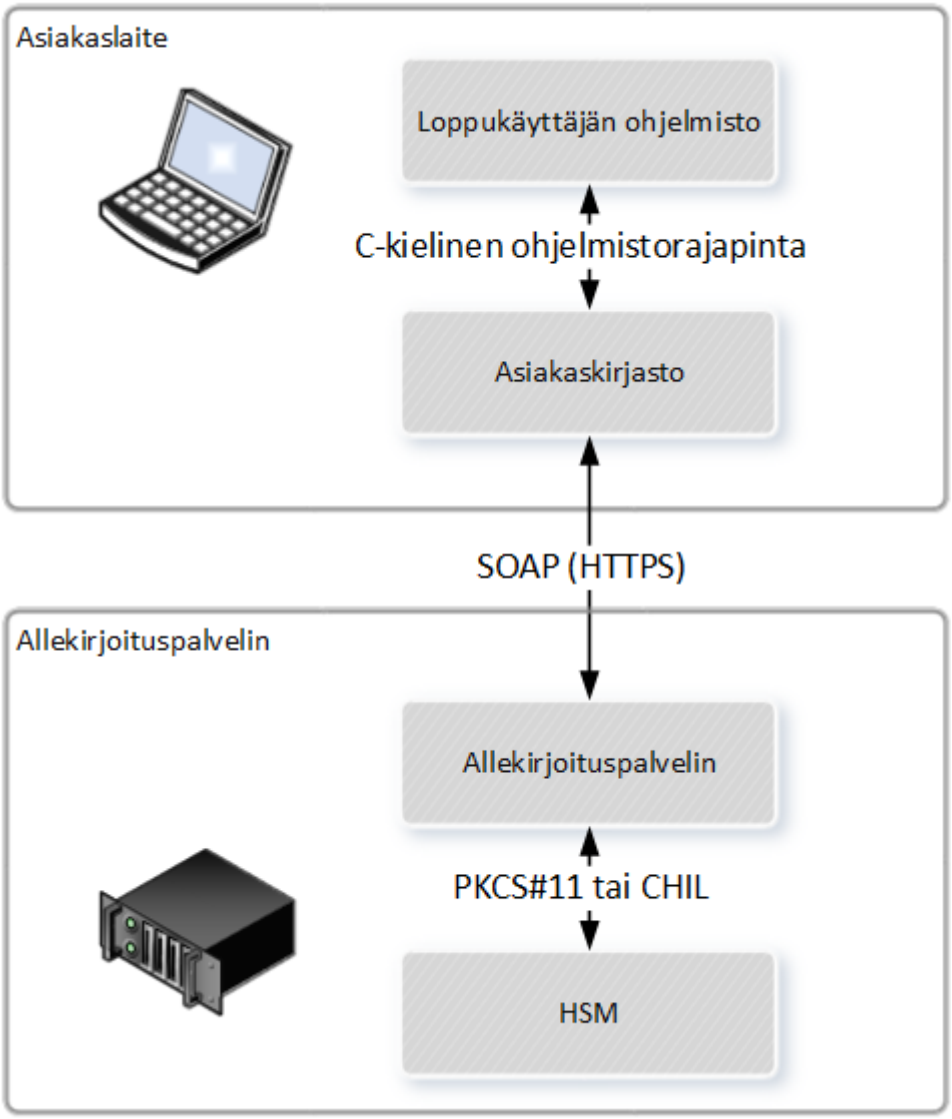
Kuva 3.7: Koko tiedoston seulonta

Näiden kahden vaihtoehdon vuoksi vastuu hajautusarvon laskemisesta vaihtelee asiakaskirjaston ja allekirjoituspalvelimen välillä. Loppukäyttäjän ei tarvitse tietää kahdesta eri vaihtoehdosta, sillä käytettävän työnkulun määrittelevät projektin asetukset järjestelmässä. Koska kyseessä on tietoturvakriittinen valinta, ei allekirjoittaja saa kyetä vaikuttamaan työnkulun valintaan. Allekirjoituspalvelin määrittää asiakaskirjastolle oman rajapintafunktionsa kautta, kumpaa työnkulkua asiakaskirjaston täytyy noudattaa.

3.6. Sovellusrajapinnat

Tiedon ja tiedostojen siirtämiseen käytettävät rajapinnat on esitetty kuvassa 3.8. Asiakaslaitteiden ja palvelimen välille valittu SOAP (lyhenne sanoista Simple Object Access Protocol) on yleisesti käytössä oleva tuotantokelpoinen HTTP-pohjainen tietoliikenneprotokolla [20]. Allekirjoituspalvelimen ohjelmisto käyttää palvelimessa olevaa HSM-laitetta kryptopalvelun läpi. Tuettuja kryptopalveluita ovat HSM:n valmistajan Thalesin

oma CHIL sekä teollisuudessa yleinen PKCS#11 [21].



Kuva 3.8: Ohjelmistorajapinnat

3.6.1. Asiakaskirjaston tarjoama sovellusrajapinta työkaluille

Asiakaskirjaston perimmäinen tarkoitus on tarjota mahdollisimman yleiskäyttöiset viestintärajapinnat järjestelmän käyttämiseen. Kirjasto tarjoaa C-kielisiä funktioita asiakasohjelmia kehittävien insinöörien käyttöön. Ohjelma 3.1 sisältää näiden C-kielisten rajapintafunktioiden määritelmät.

Parametrit `spid`, `key_id` ja `content_id` määrittävät sen, mihin allekirjoituskonfiguraatioon kutsu linkittyy. Kaikissa funktioissa esiintyvä config_filename-parametri on viittaus asiakaskirjaston omaan konfiguraatiotiedostoon, jossa määritellään esimerkiksi allekirjoituspalvelimen URL-osoite.

Ohjelma 3.1: Asiakaskirjaston tarjoamat rajapintafunktiot

```
/**
 * sign_with_hdr: allekirjoittaa tietoa, otsake erikseen
 *
 * @spid[in]: projektin SPID-tunniste
 * @key_id[in]: allekirjoitusavaimen tunniste
 * @content_id[in]: komponentin tunniste
 * @hdr[in]: osoitin tiedoston otsakkeeseen
 * @hdr_len[in]: otsakkeen koko
 * @payload[in]: osoitin kuormaan
 * @payload_len[in]: kuorman koko
 * @signature[out]: tuotettu allekirjoitus paluuarvona
 * @signature_len[out]: tuotetun allekirjoituksen koko
 * @public_key[out]: avainparin julkinen avain paluuarvona
 * @public_key_len[out]: julkisen avaimen koko
 * @config_filename[in]: asiakaskirjaston
     konfiguraatiotiedosto
 * @pkcs12_pwd[in]: pkcs#12-tiedoston salasana
 * @pkcs12_filename[in]: pkcs#12-tiedosto tunnistautumiseen
 * @return_desc[out]: virhekoodi
*/
int sign_binary(
  const char * spid,
  unsigned int * key_id,
  unsigned int * content_id,
  const unsigned char * hdr,
  const unsigned int * hdr_len,
  const unsigned char * payload,
  const unsigned int * payload_len,
  unsigned char * signature,
  unsigned int * signature_len,
  unsigned char * public_key,
  unsigned int * public_key_len,
```

```
32    const char * config_filename,
      char * pkcs12_pwd,
34    const char * pkcs12_filename,
      int return_desc);
36
  /**
38  * sign_binary: allekirjoittaa tietoa, otsake samassa
    *
40  * @spid[in]: projektin SPID-tunniste
    * @key_id[in]: allekirjoitusavaimen tunniste
42  * @content_id[in]: komponentin tunniste
    * @data[in]: osoitin allekirjoitettavaan tietoon
44  * @data_len[in]: allekirjoitettavan tiedon koko
    * @signature[out]: tuotettu allekirjoitus paluuarvona
46  * @signature_len[out]: tuotetun allekirjoituksen koko
    * @public_key[out]: avainparin julkinen avain paluuarvona
48  * @public_key_len[out]: julkisen avaimen koko
    * @config_filename[in]: asiakaskirjaston
       konfiguraatiotiedosto
50  * @pkcs12_pwd[in]: pkcs#12-tiedoston salasana
    * @pkcs12_filename[in]: pkcs#12-tiedosto tunnistautumiseen
52  * @return_desc[out]: virhekoodi
  */
54 int sign_binary(
      const char * spid,
56    unsigned int * key_id,
      unsigned int * content_id,
58    const unsigned char * data,
      const unsigned int * data_len,
60    unsigned char * signature,
      unsigned int * signature_len,
62    unsigned char * public_key,
      unsigned int * public_key_len,
```

```
64    const char * config_filename,
      char * pkcs12_pwd,
66    const char * pkcs12_filename,
      int return_desc);
68
    /**
70   * fetch_public_key: noutaa julkisen avaimen
     *
72   * @spid[in]: projektin SPID-tunniste
     * @key_id[in]: allekirjoitusavaimen tunniste
74   * @public_key[out]: avainparin julkinen avain paluuarvona
     * @public_key_len[out]: julkisen avaimen koko
76   * @config_filename[in]: asiakaskirjaston
        konfiguraatiotiedosto
     * @pkcs12_pwd[in]: pkcs#12-tiedoston salasana
78   * @pkcs12_filename[in]: pkcs#12-tiedosto tunnistautumiseen
     * @return_desc[out]: virhekoodi
80  */
    int fetch_public_key(
82    const char * spid,
      unsigned int * key_id,
84    unsigned char * public_key,
      unsigned int * public_key_len,
86    const char * config_filename,
      char * pkcs12_pwd,
88    const char * pkcs12_filename,
      int return_desc);
90
    /**
92   * fetch_user_permissions: noutaa allekirjoitusoikeudet
     *
94   * @permissions[out]: json-muotoinen lista oikeuksista
     * @config_filename[in]: asiakaskirjaston
```

```
        konfiguraatiotiedosto
 96   * @pkcs12_pwd[in]: pkcs#12-tiedoston salasana
      * @pkcs12_filename[in]: pkcs#12-tiedosto tunnistautumiseen
 98   * @return_desc[out]: virhekoodi
     */
100  int fetch_user_permissions(
       char ** permissions,
102    const char * config_filename,
       char * pkcs12_pwd,
104    const char * pkcs12_filename,
       int return_desc);
```

3.6.2. Allekirjoituspalvelimen tarjoama rajapinta asiakaskirjastolle

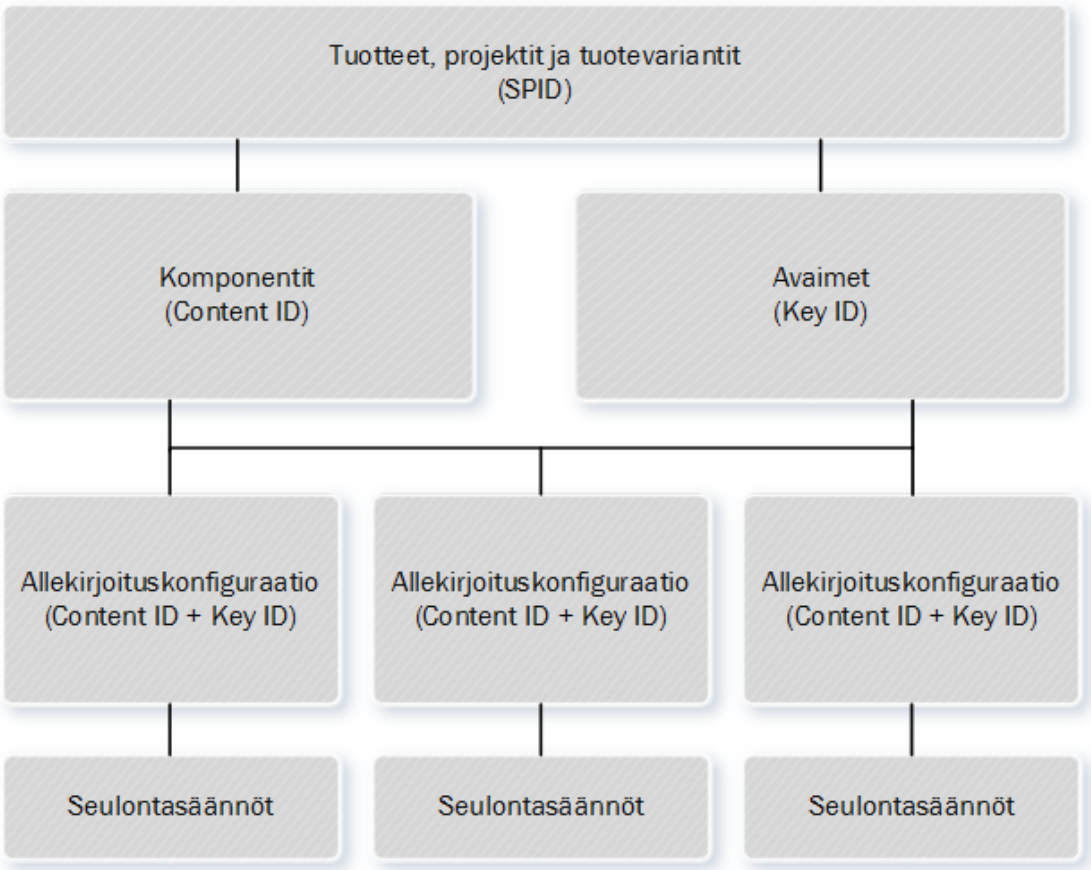
Allekirjoituspalvelimen tarjoama rajapinta voidaan linkittää varsin suoraan asiakaskirjaston tarjoamiin rajapintoihin. Palvelimen tapauksessa kyseessä on SOAP-rajapinta, joka tarjoaa seuraavat palveluoperaatiot:

- `signBinary`: käsittelee allekirjoituspyynnöt.
- `getPublicKey`: noutaa pelkän julkisen avaimen annettujen parametrien perusteella.
- `getUserPermissions`: noutaa listan kaikista oikeutetun käyttäjän allekirjoitusoikeuksista.
- `isPayloadNeeded`: määrittää allekirjoituskirjastolle, onko allekirjoituskonfiguraatiossa aktivoitu kuorman lähettäminen.

isPayloadNeeded on palveluoperaatio, joka on tehty ainoastaan asiakaskirjaston ja allekirjoituspalvelimen väliseen viestintään. Asiakaskirjasto kutsuu tätä palvelua aina ennen varsinaisen allekirjoituskutsun lähettämistä palvelimelle, ja se määrittää käytettävän työnkulun.

3.7. Tietomalli

Kuva 3.9 esittää palvelimille asetettavien konfiguraatioiden tietomallia. Sovellusylläpitäjä asettaa kyseiset konfiguraatiot kullekin projektille tietoturva-arkkitehtien määrittelemän avaintenkäyttöpolitiikan mukaan. SPID on projektin heksadesimaalimuotoinen tunniste. Tuotteiden SPID-tunnisteita hallinnoidaan keskitetysti Intelin sisäisessä järjestelmässä, joten se tulee ylläpitäjille valmiina.



Kuva 3.9: Konfiguraatioiden tietomalli

Projektille asetetaan joukko allekirjoitusavaimia ja allekirjoitettavia komponentteja. Avaimille annetaan projektin sisällä kokonaislukumuotoiset tunnisteet (Key ID), joita allekirjoittajat käyttävät pyynnön parametreinä. Järjestelmä yhdistää tunnisteen varsinaiseen avaimeen, joka on HSM:llä salattuna. Komponentin tunniste (Content ID) on 32-bittinen lukuarvo, jota käsitellään järjestelmässä 8-tavun heksadesimaaliesityksenä. Kyseinen tunniste kirjoitetaan yleensä myös allekirjoitettavan tiedoston otsakkeeseen. Allekirjoituskonfiguraatiot ovat komponenttien (Content ID) ja avainten (Key ID) tavoin

projektille (SPID) asetettuja attribuutteja.

3.7.1. Allekirjoituskonfiguraatiot

Allekirjoituskonfiguraatiot ovat komponentin ja allekirjoitusavaimen yhdistelmiä, joihin käyttäjät hakevat allekirjoitusoikeuksia. Ne ovat komponentti-avain -sidoksen osalta uniikkeja eli niitä voi olla enintään yhtä paljon kuin projektille asetettujen komponenttien ja avainten eri yhdistelmien mahdollinen lukumäärä. Tämä täyttää vaatimuksen, jossa allekirjoitusoikeuksia ei haluttu jakaa avain- tai komponenttikohtaisesti, vaan haluttiin rajata ne tiettyihin käyttötapauksiin. Jokaisella allekirjoituskonfiguraatiolla on taulukon 3.4 mukaiset attribuutit.

Taulukko 3.4: Käyttötapauskonfiguraatioiden rakenne

Attribuutti	Selite
Kuvaus	Ihmisen luettava kuvaus käyttöliittymää varten.
Content ID	Tunniste osoittaa komponenttiin projektille asetettujen komponenttien joukosta.
Key ID	Tunniste osoittaa avaimeen projektille asetettujen avaimien joukosta.
Seulontasäännöt	Lista seulontaskriptejä, jotka suoritetaan, kun allekirjoituskutsu ohjautuu kyseiseen konfiguraatioon.
Loginkirjoitustaso	Määrittää auditointilokin tason kyseiselle käyttötapaukselle. Vaihtoehdot ovat "All"(kaikki tapahtumat) tai "Failures only"(vain virheet).
Kuorman lähettämisen pakottaminen	Päälle/pois-valinta, joka määrittää, vaatiiko allekirjoituspalvelin asiakaskirjastoa lähettämään koko kuorman vai ainoastaan sen hajautusarvon. Kuorma halutaan palvelimelle siinä tapauksessa, jos se halutaan tarkastaa seulontaskripteillä.

Mikäli asiakastiimi ei halua asettaa useita eri komponentteja ja/tai avaimia projektiinsa, tukee tietomalli myös ns. villiä korttia allekirjoituskonfiguraation komponenttiparametrissa. Jos tämä on käytössä, riippumatta allekirjoituskutsun yhteydessä toimitetusta komponenttitunnisteesta (Content ID), pyyntö linkittyy aina kyseiseen allekirjoituskonfiguraatioon. Tätä käytetään usein yksinkertaisissa projekteissa, joissa allekirjoitetaan aina vain yhtä komponenttia yhdellä avaimella.

3.8. Seulonnan toteutus

Saapuvan tiedoston seulonta on toteutettu laittamalla allekirjoituspalvelimen ohjelmistoon Lua-tulkki. Lua valittiin, koska se on kevyt mutta tehokas tulkattava skriptikieli, joka on helppo sulauttaa muuhun ohjelmistoon [22].

Lua-skriptit ladataan järjestelmään hallintapalvelimien kautta ja säilötään paikallisesti json-tiedostoihin allekirjoituspalvelimilla. Seulontasäännöt asetetaan käyttötapauskonfiguraatioille ja niitä voi olla useita, jolloin ne suoritetaan peräkkäin.

Seulontajärjestelmässä on määritetty vakiot `ACCEPT` ja `DENY`, joita skripteissä voidaan käyttää suoraan paluuarvoina. Nämä paluuarvot määräävät, toteutuuko allekirjoitusoperaatio:

- Jokaisen skriptin on palautettava joko `ACCEPT` (hyväksyy) tai `DENY` (hylkää).
- Jos paluuarvoa ei ole tai se on jokin muu, seulontajärjestelmä palauttaa `DENY`.
- Jos Lua-tulkki aiheuttaa keskeytyksen, seulontajärjestelmä palauttaa `DENY`.
- Jokaisen käyttötapauskonfiguraatiolle asetetun skriptin täytyy palauttaa `ACCEPT`, jotta allekirjoitus suoritetaan.

Seulontajärjestelmä tarjoaa siinä ajettaville Lua-skripteille ohjelmistorajapintoja tiedoston lukemiseen ja kirjoittamiseen. Näiden funktioiden operaatiot suoritetaan seulontajärjestelmän C-kielisen koodin toimesta suoraan seulottavalle tiedostolle ja tulos palautetaan Lua-skriptille. Sekä lukemiseen, että kirjoittamiseen on neljä rajapintafunktiota.

Tiedon lukemiseen käytettävät funktiot ovat:

- `get_token_bit(byte_index, bit_index)`: Lukee bitin tiedostosta hyödyntäen tavujen indeksointia. Esimerkiksi `get_token_bit(7, 3)` hakee seitsemännen tavun kolmannen bitin.
- `get_token_byte(byte_index)`: Lukee tavun tiedostosta. Esimerkiksi `get_token_byte(5)` hakee tiedoston viidennen tavun.
- `get_token_byte_w(word_index, byte_index)`: Lukee tavun tiedostosta hyödyntäen sanojen indeksointia. Esimerkiksi `get_token_byte_w(5, 2)` hakee viidennen sanan toisen tavun.

- `get_token_word_w(word_index)`: Lukee sanan tiedostosta. Esimerkiksi `get_token_word_w(2)` hakee toisen sanan.

Tiedon kirjoittamiseen käytettävät funktiot ovat:

- `set_token_bit(byte_index, bit_index, value)`: Asettaa bitin hyödyntäen tavujen indeksointia. Esimerkiksi `set_token_bit(3, 2, 1)` asettaa kolmannen tavun toisen bitin arvoksi 1.

- `set_token_byte(byte_index, value)`: Asettaa tavulle arvon. Esimerkiksi `set_token_byte(4, 0x24)` asettaa neljännen tavun arvoksi 0x24.

- `set_token_byte_w(word_index, byte_index, value)`: Asettaa tavulle arvon hyödyntäen sanojen indeksointia. Esimerkiksi `set_token_byte_w(4, 2, 0x24)` asettaa neljännen sanan toisen tavun arvoksi 0x24.

- `set_token_word_w(word_index, value)`: Asettaa sanalle arvon. Esimerkiksi `set_token_word_w(2, 0x12345678)` asettaa toisen sanan arvoksi 0x12345678.

3.8.1. Turvallisuus

Seulontajärjestelmä mahdollistaa ulkopuolisen ohjelmakoodin lataamisen palvelimelle. Sen vuoksi järjestelmän on oltava suojattu tätä kautta tulevien hyökkäysten varalta. Allekirjoitusjärjestelmän turvallisuuden vuoksi seulontajärjestelmälle on määritetty käsittelyt seuraaville erikoistapauksille:

- Sana on aina 32 bittiä (myös 64-bittisissä järjestelmissä).

- Tavu on aina 8 bittiä.

- Get-funktiot eli lukufunktiot palauttavat -1, jos annetut parametrit osoittavat tietopaketin ulkopuolelle (engl. out of range index).

- Set-funktiot eli kirjoitusfunktiot, jotka yrittävät kirjoittaa tietopaketin ulkopuolelle (engl. out of range index), eivät tee mitään.

- Set-funktiot eli kirjoitusfunktiot ylikirjoittavat käyttämättömät bitit nollaksi. Esimerkiksi `set_token_byte(1, 1)` asettaa ensimmäisen tavun arvoksi 0x0001.

- Liian suuret syötteet katkaistaan maksimikokoon. Esimerkiksi `set_token_byte(1, 0xFFEE)` asettaa ensimmäisen tavun arvoksi 0xEE.

Tämän lisäksi skriptit ajetaan rajatussa ympäristössä eikä niille anneta käyttöön kuin pienin mahdollinen määrä Luan standardikirjastoja. Yksikään näistä ei tarjoa suoria rajapintoja tietoturvakriittisiin käyttöjärjestelmän palveluihin kuten tiedostojärjestelmään.

3.8.2. Esimerkkejä

Alla on esitetty kaksi yksinkertaistettua esimerkkiä tyypillisistä käyttötapauksista. Ohjelma 3.2 on esimerkki yksinkertaisesta skriptistä, joka tarkistaa, että otsakkeessa on sama komponenttitunniste kuin minkä allekirjoittaja on pyynnössään antanut content_id-parametrina. Skripti olettaa, että komponenttitunniste on otsakkeen viides 32-bittinen sana, kuten A-otsakkeessa. Skriptin tehtävä on estää yritykset allekirjoittaa mitään muuta kuin allekirjoituskonfiguraatiossa määrättyä komponenttia kyseisellä avaimella.

Ohjelma 3.2: Esimerkkiskripti tiedon lukemiseen

```
-- Esimerkki-skripti sisallon lukemiseen
do
  -- Lue viides 32-bittinen sana A-otsakkeesta
  komponenttitunniste = get_token_word_w(5)
  if komponenttitunniste == 0x1234ABCD then
    return ACCEPT -- Haluttu arvo
  end
  return DENY -- Arvo ei kelpaa
end
```

Ohjelma 3.3 on esimerkkiskripti, joka asettaa aikaleiman allekirjoitettavan tiedoston alkuun. Olemassa oleva tieto ensimmäisessä sanassa ylikirjoitetaan.

Ohjelma 3.3: Esimerkkiskripti tiedon kirjoittamiseen

```
-- Esimerkki-skripti aikaleiman kirjoittamiseen
do
  -- Aseta aikaleima tiedoston alkuun
  set_token_word_w(1, os.time())
  return ACCEPT -- Haluttu arvo
```

```
6 end
```

3.9. Järjestelmän yleinen tietoturva

Tietoturvan osalta Intelin omat tietoturvapolitiikat ja -käytännöt ohjasivat toteutusta merkittävästi. Esimerkiksi kaiken ohjelmistokoodin piti noudattaa ohjesääntöjä ja läpäistä yrityksen määrittelemät katselmointiprosessit. Teknisen tietoturvan lisäksi tietoturvapolitiikat ohjasivat myös hallinnollisen tietoturvan toteuttamista.

3.9.1. Fyysinen turvallisuus

Yhteenkään tuotantopalvelimeen ei saa etäyhteyttä, vaan järjestelmäylläpitäjän on aina kirjauduttava paikallisesti. Palvelimet ovat lukituissa kaapeissaan Intelin omissa datakeskuksissa, ja kaappien avaimet ovat toisilla järjestelmäylläpitäjillä. Henkilö, jolla on paikallinen tunnus palvelimelle, ei saa pitää hallussaan kyseisen palvelinkaapin avainta.

Allekirjoitusavaimet on salattu Thalesin HSM:ien avulla. HSM:t ovat laajennuskortteina palvelimissa, ja siten ne ovat myös lukituissa kaapeissa. Niissä on myös kehittyneet sisäänrakennetut mekanisminsa Side Channel -hyökkäysten varalle. HSM-laitteiden konfiguraatioiden muuttamiseen ja alustamiseen määriteltiin omat prosessinsa ja niiden suorittamiseen tarvitaan järjestelmäylläpitäjien lisäksi aina kaksi ylläpitokortin haltijaa, joilla on toistensa korttien salasanat. Jokaisessa HSM:ssä on kytkettynä kortinlukija.

Kaikkien läsnäolleiden allekirjoittama kirjallinen raportti on tehtävä aina, kun palvelinkaappi avataan. Lisäksi palvelinhuone on lukittu ja sillä on oma kulunvalvontansa sekä tallentava valvontajärjestelmä.

3.9.2. Neljän silmän periaate

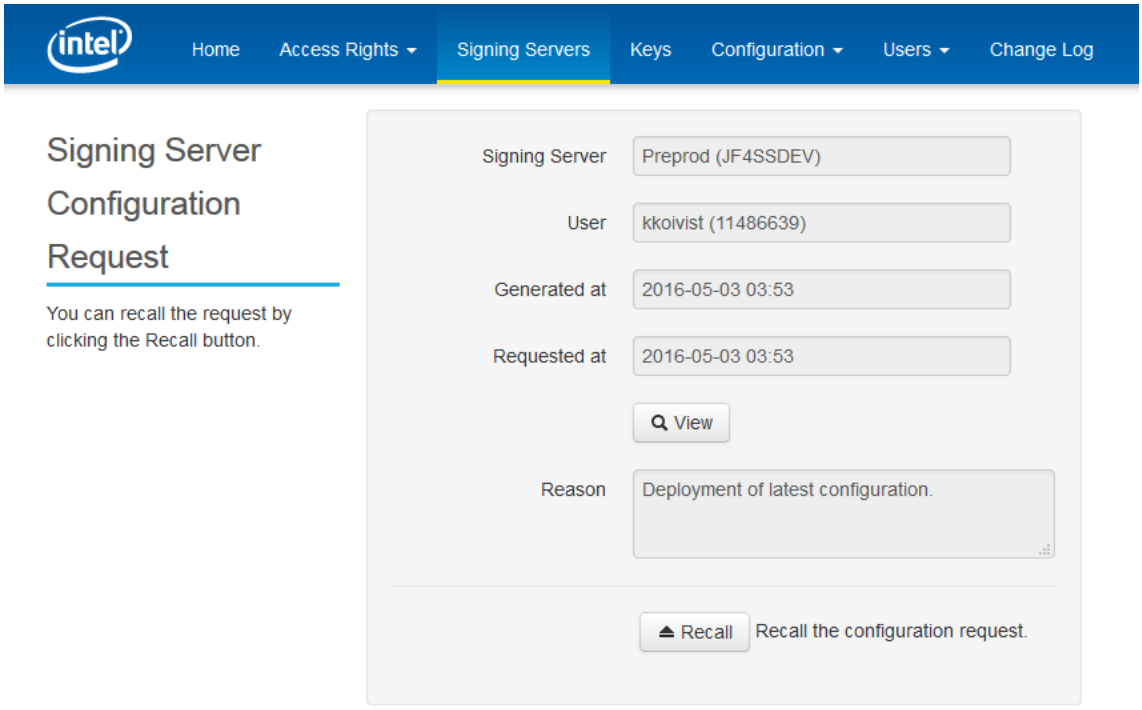
Neljän silmän periaatteella tarkoitetaan toimintatapaa, jossa toiminnan suorittamiseen tarvitaan samanaikaisesti kaksi henkilöä. Tämän käytännön tarkoituksena on ehkäistä yksittäisen henkilön tekemiä väärinkäytöksiä järjestelmän kriittisissä osissa. Neljän silmän periaatetta sovelletaan käytännössä toteutetun järjestelmän kolmessa kohdassa:

- Palvelimille fyysisesti kirjautumiseen.
- Sovellusylläpitäjien toimiin.

- HSM-laitteiden hallintaan.

Hallinta-, tietokanta- ja allekirjoituspalvelimet ovat järjestelmän kriittisiä elementtejä, sillä ne säilövät tärkeää käyttöoikeuksiin liittyvää tietoa, allekirjoituskonfiguraatioita ja käyttävät suoraan HSM:iä. Palvelinten kovalevyillä on myös muun muassa niiden lokitiedostoja sekä varmenteet. Lisäksi ylläpito-oikeudet omaava käyttäjä saa käyttöönsä kaikki palvelimen käyttöjärjestelmän tarjoamat resurssit. Tämän vuoksi palvelimelle kirjautumiseen tarvitaan aina vähintään kaksi henkilöä, joista toisella on palvelinkaapin avain ja toisella paikallinen käyttäjätunnus palvelimelle. Molemmat henkilöt hyväksyvät kaikki tehdyt toimenpiteet ja vahvistavat ne itse allekirjoittamillaan paperisilla raporteilla.

Järjestelmän kannalta kriittisiä konfiguraatioita, kuten allekirjoitusavainten linkityksiä ja allekirjoitusoikeuksia, hallitaan web-käyttöliittymän kautta. Tämän vuoksi toisen sovellusylläpitäjän on hyväksyttävä uudet allekirjoituskonfiguraatiot järjestelmässä, jotta sovellusylläpitäjän tai SPID-isännän tekemät muutokset astuvat voimaan (kuva 3.10). Lisäksi sovellusylläpitäjä tai SPID-isäntä ei voi myöntää itselleen allekirjoitusoikeuksia, vaan heidän tekemänsä allekirjoitusoikeuspyynnöt menevät hyväksyttäväksi muille sovellusylläpitäjille.



Kuva 3.10: Sovellusylläpitäjä ei pysty hyväksymään omia muutoksiaan järjestelmään

Kolmantena kohtana on itse HSM:t. Kaikki ylläpito- ja konfiguraatiotoimenpiteet tehdään asennuksen yhteydessä luotujen toimikorttien avulla. Toimikortit on suojattu salasanoilla ja jaettu ristiin niin, että kortin haltijalla on korttiparin toisen kortin salasana ja vastaavasti toisen kortin haltijalla on ensimmäisen kortin salasana. Yhteensä toimikortteja on useampi pari, mutta yksi pari riittää operointiin.

3.9.3. Käyttöoikeudet ja tunnistautuminen

Saatavuuden parantamiseksi allekirjoituspalvelinten haluttiin toimivan itsenäisesti ilman jatkuvaa yhteyttä hallinta- ja tietokantapalvelimiin. Tämän teki mahdolliseksi paikalliset konfiguraatio- ja käyttöoikeustiedot, jotka palvelimille tallennettiin. Näin ollen allekirjoituspalvelimet tietävät itse, kenellä on oikeudet allekirjoittaa mitäkin. Käyttöoikeuslistojen muuttuessa ne lähetetään uudelleen hallintapalvelimilta allekirjoituspalvelimille.

Varsinainen käyttäjien tunnistautuminen tapahtuu henkilökohtaisten varmenteiden avulla. Nämä varmenteet myöntää Intelin PKI-palvelu, johon käyttäjät kirjautuvat Intelin yleisillä käyttäjätunnuksillaan. Varmenne on käyttäjän salasanalla suojattuna PKCS#12-säiliössä, joka annetaan asiakaskirjastolle yhtenä parametrina. Varmenteet valittiin, koska haluttiin yhdellä tunnistautumismekanismilla täyttää vaatimukset, jotka liittyvät automaattisiin käännöspalvelimiin sekä usean käyttöjärjestelmän tukemiseen. Käyttämällä varmenteita käyttäjien ei tarvitse tallentaa yleisiä käyttäjätunnuksiaan automatisoidakseen allekirjoitusprosessit.

3.9.4. Lokitiedostot

Logitiedostot ovat muun muassa auditoinnin ja kiistämättömyyden vuoksi tärkeitä tietolähteitä. Ne ovat selkeä kohde hyökkääjille, jotka haluavat peittää omat jälkensä murtauduttuaan järjestelmään. Lisäksi lokitiedostoissa on tietoa, jota ei saa antaa ulkopuolisille. Esimerkki tällaisesta tiedosta on kriittisten tuotantokomponenttien allekirjoittajien tunnisteet, joita hyökkääjä voi käyttää esimerkiksi valikoidakseen kohteita sosiaaliseen manipulointiin.

Ainoat henkilöt, joilla on lukuoikeus lokitiedostoihin, ovat sovellus- ja järjestelmäylläpitäjät. Kirjoitusoikeudet ovat ainoastaan allekirjoituspalvelimilla ajettavilla prosesseilla, joita puolestaan voidaan hallita vain kirjautumalla palvelimelle paikallisesti neljän silmän

periaatetta noudattamalla.

Lisäksi lokitiedostojen manipulointia ehkäistään kopioimalla niitä ristiin eri palvelinten välillä. Vain uudet syntyneet lokitiedot kopioidaan - ei mahdollisesti jälkikäteen muutettuja tietoja. Näin ollen hyökkääjän on kyettävä poistamaan jälkensä jokaisesta kopiosta eri palvelimilla peittääkseen jälkensä.

3.9.5. Tietoturvapolitiikat ja -prosessit

Järjestelmäylläpito- ja infrastruktuuritehtävät annettiin Intelin IT-yksikön hoidettavaksi. Yksikkö on velvoitettu noudattamaan yhtiön tietoturvapolitiikkaa, mikä kattaa muun muassa järjestelmäpäivitykset sekä varmuuskopiot.

Tämän lisäksi määritettiin ja dokumentoitiin erinäisiä tarkentavia ohjeistuksia ja prosesseja esimerkiksi HSM:ien käsittelyyn, järjestelmän auditointiin ja onnettomuudesta palautumiseen (engl. disaster recovery). Nämä prosessit tehtiin yhteistyössä tietoturva-asiantuntijoiden kanssa. Ne menevät teknisen toteutuksen ulkopuolelle, eikä niitä sen vuoksi käydä tässä työssä tarkemmin läpi.

3.10. Asiakasohjelmat

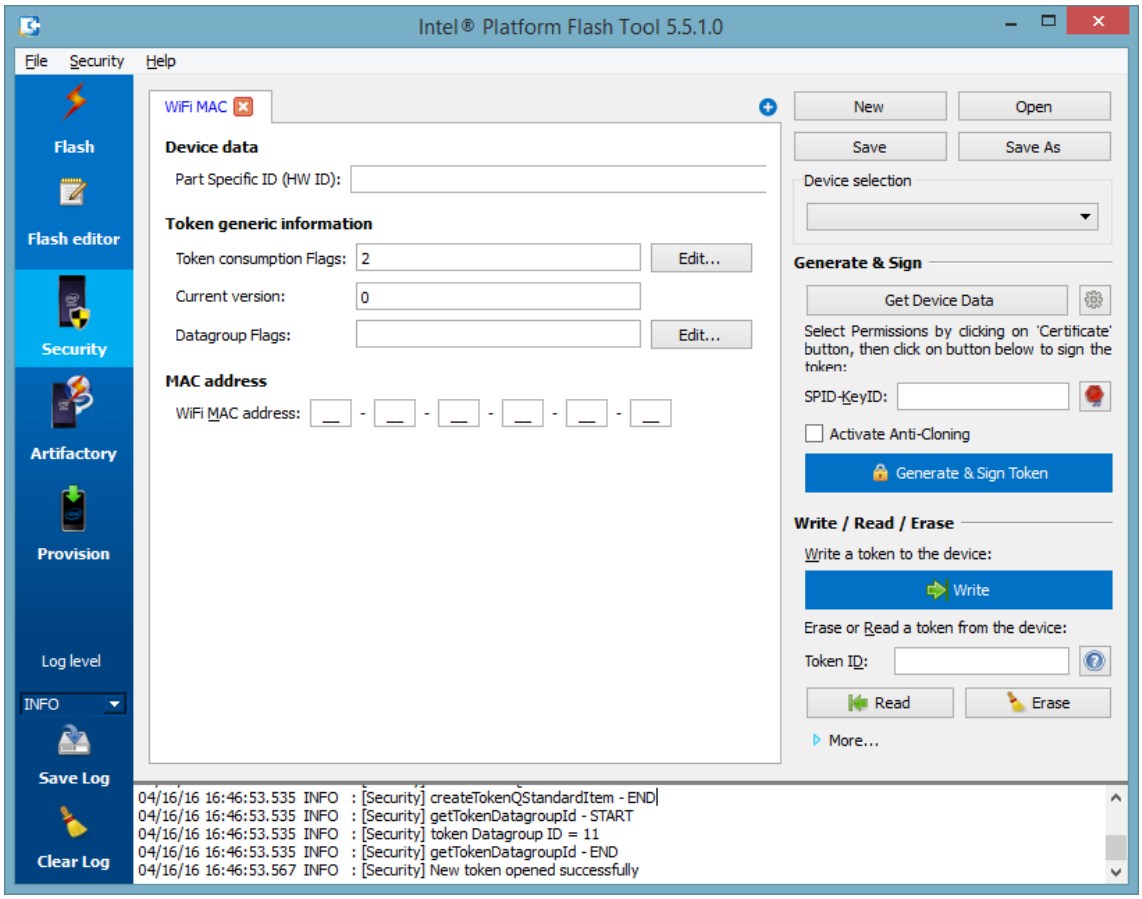
Asiakasohjelmat ovat tietokoneohjelmia, jotka tarjoavat käyttäjärajapinnan allekirjoitusjärjestelmään. Ne ovat suurimmaksi osaksi Intelin sisäisten työkalutiimien toteuttamia. Osa työkaluista on Intelin omaan käyttöön ja osa sekä Intelin että sen sopimusvalmistajien käyttöön. Dynaamisen asiakaskirjaston tarjoamien korkean tason allekirjoitusfunktioiden ansiosta työkalutiimit pystyvät keskittymään työkalujensa ydintoiminnallisuuksien kehittämiseen. Erot eri työkalujen välillä syntyvät eri tuotekehitystiimien käyttötapauksista ja niiden käyttämien ohjelmistokomponenttien ja tiedostojen otsakkeista. Allekirjoittaminen on vain yksi näiden työkalujen toiminnallisuuksista. Merkittävimmät toteutettua allekirjoitusjärjestelmää hyödyntävät asiakasohjelmat on listattu taulukkoon 3.5.

Kuva 3.11 on kuvakaappaus Token Manager Tool -työkalusta. Yksi työkalulla luotavista konfiguraatiopaketeista on kuvassa esiintyvä WiFi MAC -paketti, jolla asetetaan laitteen WLAN-moduulin parametrit ja eritoten MAC-osoite. WLAN on lyhenne englannin kielen sanoista Wireless Local Area Network, joka voidaan vapaasti suomentaa langattomaksi paikalliseksi verkoksi. MAC-osoite puolestaan on lyhenne englannin kielen

Taulukko 3.5: Esimerkkejä asiakasohjelmista

Työkalu	Käyttökohde	Käyttötarkoitus
Token Manager Tool	Mobiilialustat	Konfiguraatiopakettien (engl. token) luominen, allekirjoittaminen ja laitteelle provisiointi.
Stitcher	Mobiilialustat	Firmware-kuvan (engl. Integrated Firmware Image eli IFWI) luominen ja siihen kuuluvien komponenttien allekirjoittaminen.
Platform Flash Tool	Mobiilialustat	Työkalu hyödyntää Token Manager Tool ja Stitcher -työkaluja. Se mahdollistaa mobiilituotteiden alustamisen ja käyttöjärjestelmä- ja firmware-kuvien provisioinnin.
CSS Signing Tool	Integroidut modeemit ja antennit	Komentorivityökalu modeemien ja antennien (esim. Bluetooth ja WLAN) firmware-komponenttien allekirjoittamiseen. Käytetään osana laajempia työkalukokonaisuuksia.

sanoista Media Access Control. Se on osoite, joka yksilöi laitteen verkossa.



Kuva 3.11: Kuvakaappaus Token Manager Tool -työkalusta

4. ARVIOINTI

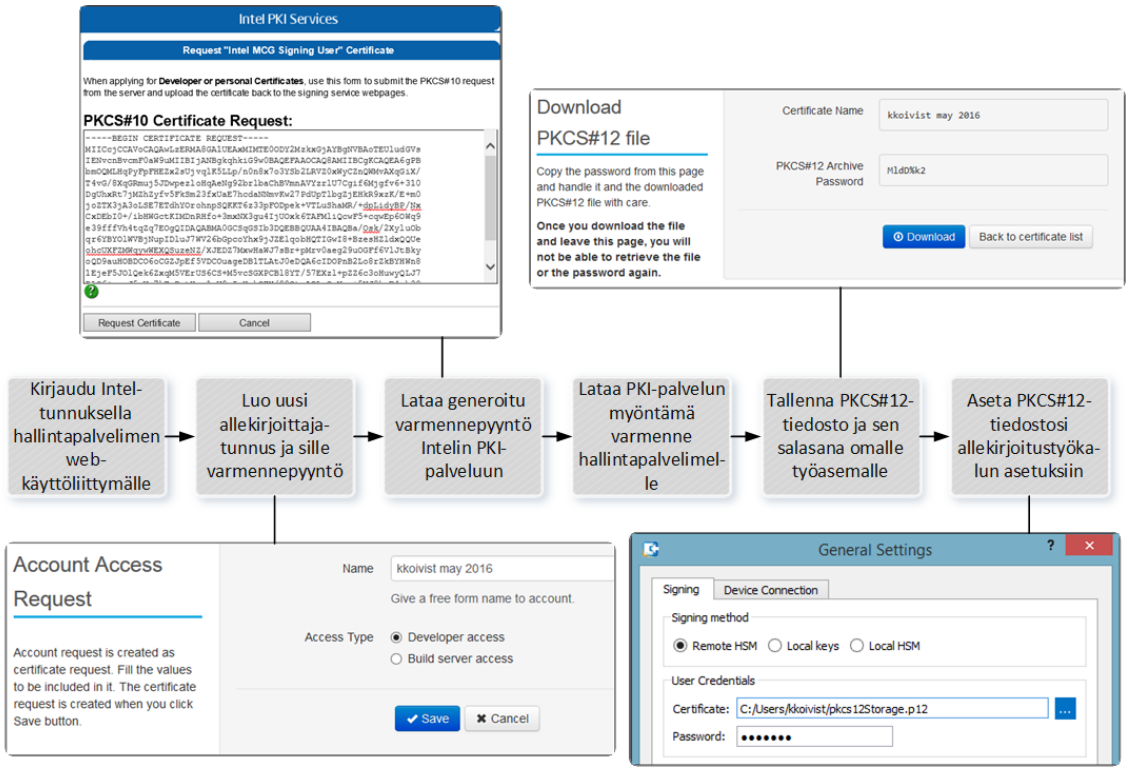
Tämän luvun kohta 4.1. käy läpi, miten toteutettu järjestelmä näkyy yksittäiselle allekirjoittajalle. Kohta 4.2. käsittelee järjestelmästä kerättyä tilastomateriaalia ja arvioi niiden perusteella järjestelmän tuotantokelpoisuutta kuormituksen suuruuden ja sen keston näkökulmasta. Kohta 4.3. arvioi seulonnan vaikutusta allekirjoitusprosessiin. Järjestelmästä tehtiin ylläpidettävyysanalyysi, joka käydään läpi kohdassa 4.4. Kohta 4.5. käy läpi järjestelmän virheensietokykyä ja saatavuutta. Kohta 4.6. esittelee lyhyesti Intelin tietoturvalautakunnan järjestelmästä tekemän tietoturva-analyysin. Lopuksi tulosten ja palautteen pohjalta kohtaan 4.7. on kerätty jatkokehitysajatuksia.

4.1. Käyttökokemus

Koska toteutettu järjestelmä ei tarjoa käyttöliittymää varsinaiseen allekirjoittamiseen, on kaikki allekirjoittamiseen liittyvä käyttökokemus riippuvainen käytettävästä allekirjoitustyökalusta. Näissä työkaluissa allekirjoitus on vain osa varsinaista tiedoston luonti- tai muokkausprosessia eikä käyttäjän tarvitse siihen kiinnittää tavallisessa käytössä juuri edes huomiota.

Allekirjoitustoimintojen käyttöönotto puolestaan vaatii käyttäjältä erinäisiä toimia liittyen toteutettuun järjestelmään. Aluksi allekirjoittajan on luotava itselleen allekirjoitustunnus. Tunnuksen luonnin vaiheet on esitetty kuvassa 4.1. Prosessi sisältää useita käsintehtäviä toimenpiteitä, koska Intelin PKI-palvelu ei tarjonnut ohjelmallisia rajapintoja varmenteiden pyytämiseen. Tunnuksen luominen on kuitenkin nopea toimenpide ja kestää noin yhdestä kahteen minuuttia. PKI-palvelu myöntää varmenteen automaattisesti hallintapalvelimen generoiman pyynnön perusteella, vaikkakin varmennepyyntö on kopioitava käsin hallintapalvelimen käyttöliittymältä PKI-palvelun web-käyttöliittymälle.

Kun allekirjoittajalla on aktiivinen allekirjoitustunnus, hän voi hallintapalvelimen web-käyttöliittymän kautta pyytää allekirjoitusoikeuksia allekirjoituskonfiguraatioihin. Oi-



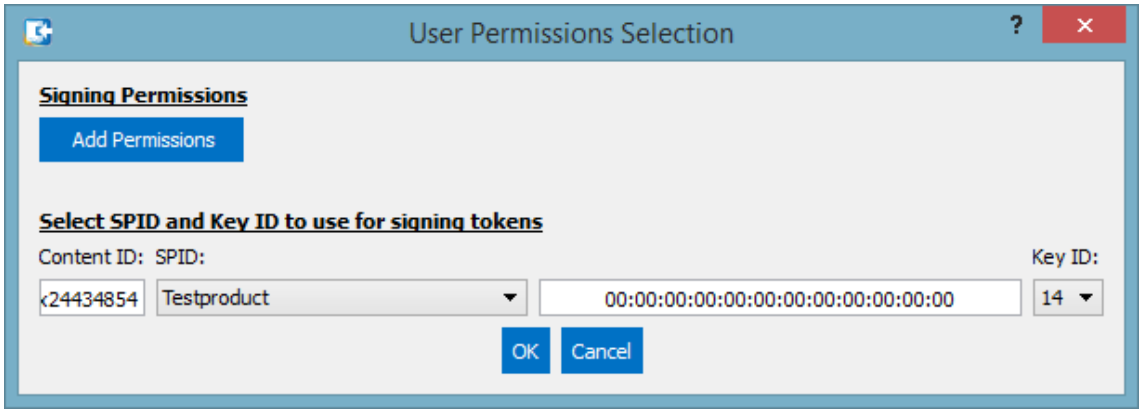
Kuva 4.1: Allekirjoittajan tunnuksen luominen

keuspyynnöt käsitellään käsin, joten oikeuksien saaminen voi kestää tyypillisesti minuuteista tunteihin. Myönnetyt allekirjoitusoikeudet ovat muutaman minuutin viiveellä käytettävissä (hallintapalvelin lähettää päivitetyt käyttöoikeuslistat allekirjoituspalvelimille automaattisesti).

Se, miten allekirjoituspyyntö käytännössä kohdistetaan tietylle allekirjoituskonfiguraatiolle, riippuu paljon allekirjoitustyökalusta. Komentorivityökalujen ja käännöspalvelimien käyttämien allekirjoitusohjelmistojen kohdalla allekirjoittajan täytyy tietää SPID-arvo sekä komponentti- ja avaintunnisteet, mitkä hän syöttää työkalulle (tyypillisesti komentorivillä tai työkalun konfiguraatiotiedostossa). Graafisissa käyttöliittymissä hyödynnetään pääsääntöisesti rajapintaa, joka hakee käyttäjän allekirjoitusoikeudet (kuva 4.2). Tämä helpottaa esimerkiksi tuotteen SPID-arvon valitsemista, kun voidaan käyttää projektien oikeita nimiä heksadesimaaliarvojen sijasta.

4.2. Käyttöaste ja sen kesto

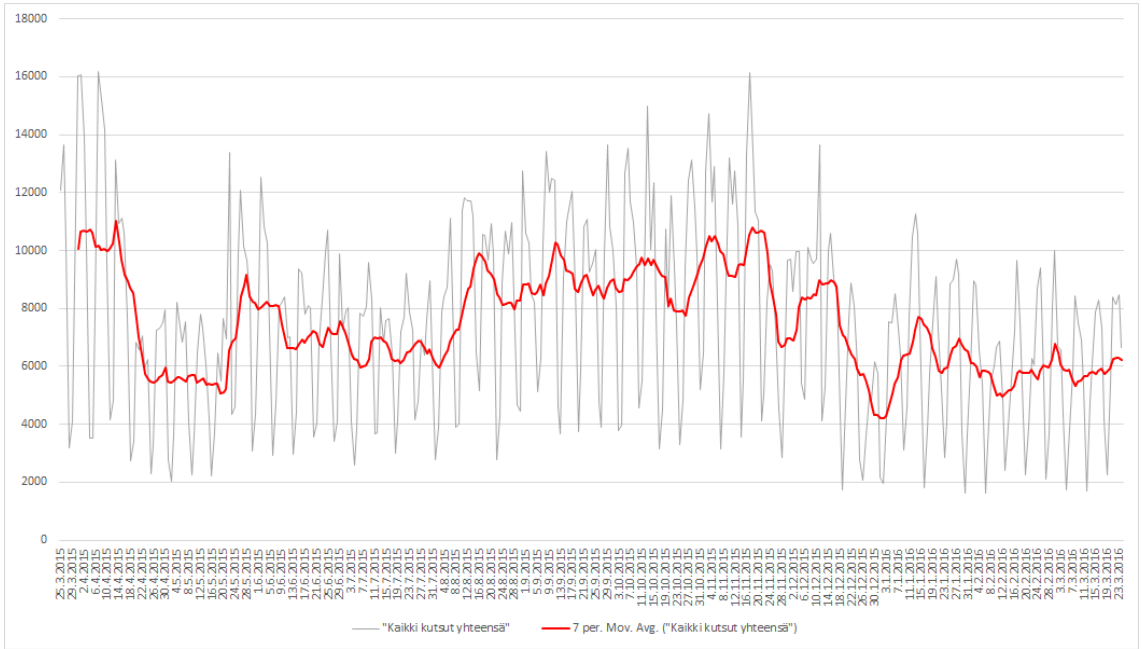
Koska järjestelmä on ollut jo tuotantokäytössä, on siitä saatavilla myös teolliseen käyttöön pohjautuvia tilastoja. Tilastot pohjautuvat tuotannon allekirjoituspalvelimien tuotta-



Kuva 4.2: Allekirjoituskonfiguraation valinta Token Manager Tool -työkalussa

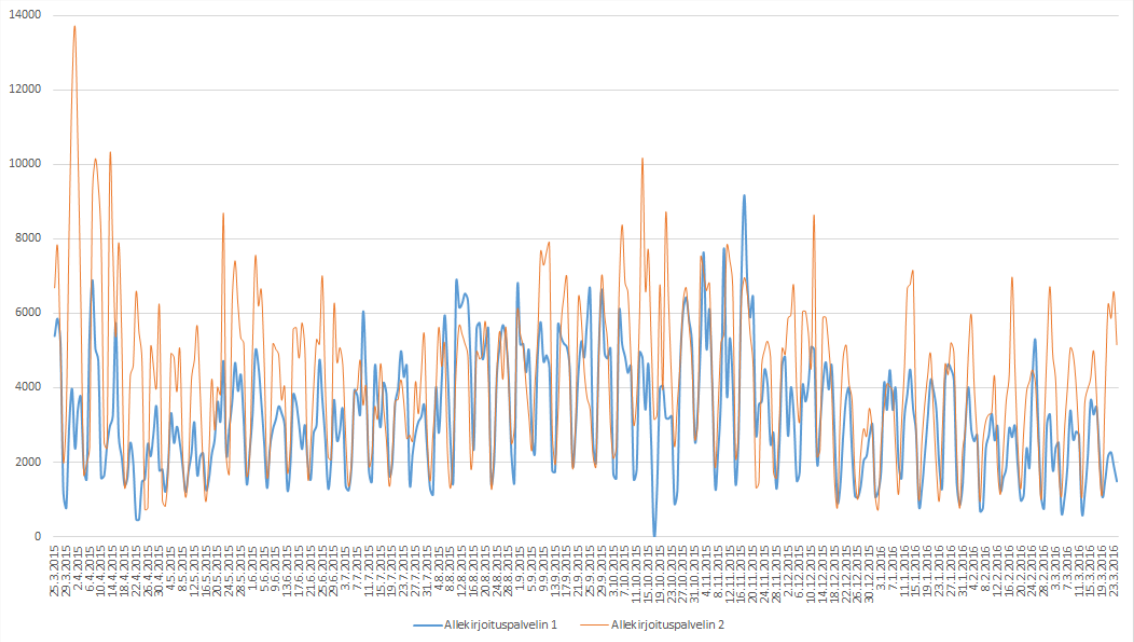
miin lokitiedostoihin viimeisen vuoden ajalta (25.3.2015-24.3.2016). Näitä palvelimia on kaksi, ja niihin viitataan nimillä Allekirjoituspalvelin 1 ja Allekirjoituspalvelin 2. Palvelimista toinen sijaitsee Pohjois-Amerikassa ja toinen Suomessa.

Kuva 4.3 näyttää allekirjoituspalvelimien kokonaiskäyttöasteen. Kuvassa on päiväkohtainen kuvaaja sekä seitsemän päivän liukuva keskiarvo. Päiväkohtaisen kuvaajan sahaava muoto aiheutuu viikonlopuista, jolloin kuormamäärät ovat merkittävästi pienemmät työntekijöiden viettäessä vapaapäiviään. Osa asiakasorganisaatioiden ja -tiimien käännös- ja integraatiopalvelimista toimii kuitenkin myös viikonloppuisin. Vaihtelevuus kokonaiskäyttöasteessa johtuu tuotteiden siirtymisistä eri vaiheisiin elinkaariaan. Esimerkiksi korkeat piikit johtuvat todennäköisesti aktiivisista tuotekehitysvaiheista.



Kuva 4.3: Kokonaiskäyttöaste

Kuvassa 4.4 on kuvaajat päiväkohtaisille pyyntömäärille. Kummallekin allekirjoituspalvelimelle on oma kuvaajansa. Lokakuun 17. päivänä vuonna 2015 Allekirjoituspalvelin 1:llä oli käyttökatko, joka selittää kuvaajan tippumisen nollaan kyseisenä ajankohtana.



Kuva 4.4: Allekirjoituspalvelimien kuormitus

Vuoden tarkastelujakson käyttöasteen yhteenveto on listattu taulukkoon 4.1. Sekä taulukosta että kuvasta 4.4 nähdään, että Allekirjoituspalvelin 2:n käyttöaste on suurempi. Tähän vaikuttaa merkittävästi ainakin kaksi tekijää. Ensinnäkin palvelinten fyysinen sijainti eri mantereilla (Pohjois-Amerikassa ja Euroopassa) aiheuttaa eroja kutsujen ohjautumisissa kuormanjakopalvelimen läpi. Toinen iso tekijä on Allekirjoituspalvelin 1:lle sattuneet ongelmat kuluvan vuoden aikana, jolloin tarkkailupalvelin joutui irrottamaan sen tuotannosta useita kymmeniä kertoja viallisen toiminnan vuoksi. Näiden katkojen kesto vaihteli minuuteista tunteihin, kunnes ylläpitäjät saivat tilanteen korjattua. Katkojen ajan kaikki kutsut ohjautuivat Allekirjoituspalvelin 2:lle.

Taulukko 4.1: Käyttöastestatistiikan yhteenveto

Palvelin	Kutsut	Onnistuneet	Virheet	Virheiden osuus
Allekirjoituspalvelin 1	1 185 163	1 184 226	937	0,079 %
Allekirjoituspalvelin 2	1 551 118	1 550 219	899	0,058 %
Yhteensä	2 736 281	2 734 445	1 836	0,067 %

Tarkastelujakson aikana tuotantoympäristön lokeihin ei tallentunut yhtäkään järjestelmävirhettä. Suurin osa tallennetuista virheistä johtui allekirjoittajan puutteellisista alle-

kirjoitusoikeuksista. Myös seulonnan hylkäämät allekirjoituspyynnöt tallentuvat lokeihin virheilmoituksina. Tapahtuneet virheet on eritelty taulukossa 4.2.

Taulukko 4.2: Tarkastelujakson virheiden erittely

Virhe	Lukumäärä
Allekirjoittajalla ei ollut vaadittavia oikeuksia.	1 795
Seulonta hylkäsi pyynnön.	38
Pyydettyä allekirjoituskonfiguraatiota ei ole olemassa.	3
Yhteensä	1 836

4.3. Seulonnan vaikutukset

Seulonnan vaikutusta allekirjoitusprosessiin allekirjoittajan näkökulmasta testattiin mittaamalla kokonaisprosessin kestoa allekirjoitustyökalulta allekirjoituspalvelimelle ja takaisin. Testiasetelmia oli kaksi, joissa molemmissa allekirjoitettiin samaa neljän kilotavun suuruista tietopakettia. Ensimmäisessä asetelmassa koko paketti lähetettiin palvelimelle, mutta se allekirjoitettiin suoraan ilman seulontaa. Toisessa asetelmassa koko paketti lähetettiin palvelimelle ja se seulottiin tuotantotapausta vastaavalla tavalla käyttäen 140 seulontaskriptiä, jotka kukin lukivat paketista 3-12 arvoa. Testit suoritettiin kaikki samalta tietokoneelta käyttäen samaa allekirjoituspalvelinta Intelin sisäverkossa. Tulokset voidaan nähdä taulukosta 4.3.

Taulukko 4.3: Seulonnan vaikutus kokonaisprosessin kestoon

	Asetelma 1 (ms)	Asetelma 2 (ms)	Erotus (ms)
Testikerta 1	394	406	12
Testikerta 2	382	377	-5
Testikerta 3	584	412	-172
Testikerta 4	614	363	-251
Testikerta 5	531	657	126
Testikerta 6	320	681	361
Testikerta 7	618	360	-258
Testikerta 8	378	389	11
Testikerta 9	313	371	58
Testikerta 10	319	672	353
Testikerta 11	331	359	28
Keskiarvo	435	459	24
Vaihteluväli	305	322	619

Tuloksista nähdään, että seulonnan vaikutus kokonaisprosessin kestoon on hyvin pieni, ja sitä voidaan pitää allekirjoittajan näkökulmasta lähes merkityksettömänä. Vaihteluvälit

kokonaiskestossa ovat suuria, mikä johtuu verkon reitityksistä sekä TLS-suojatun yhteyden muodostamisesta, joka siis tapahtui jokaisen kutsun kohdalla uudelleen.

Seulonnan käyttöönotto on mahdollistanut kriittistenkin allekirjoitusoikeuksien antamisen useammalle henkilölle. Tämä on vähentänyt niiden tuotekehitysinsinöörien työtaakkaa, joilla alunperin olivat kyseiset allekirjoitusoikeudet. Myös reagointiaika sopimusasiakkaiden tarpeisiin on tämän vuoksi lyhentynyt.

4.4. Ohjelmakoodin ylläpidettävyysanalyysi

Ylläpidettävyysanalyysi (engl. maintainability analysis) arvioi ohjelmiston ylläpidettävyyttä ja jatkokehitysmahdollisuuksia riippumatta siitä, mikä taho ohjelmistoa ylläpitää tai jatkokehittää. Intelin asiantuntija teki analyysin allekirjoituspalvelimen ohjelmistolle [23]. Analyysiin kuuluu kaksi osaa: allekirjoituspalvelimen ohjelmakoodin statistiikka ja syklomaattinen monimutkaisuus. Ohjelmakoodin statistiikka antaa pintapuolisen yleiskuvan ja ensivaikutelman ohjelmiston koosta ja monimutkaisuudesta. Syklomaattinen monimutkaisuus mittaa lineaarisesti itsenäisiä polkuja ohjelmakoodissa. Se toimii kelvollisena mittarina ohjelmiston monimutkaisuudesta: mitä suuremman monimutkaisuusarvon ohjelmakoodi saa, sen vaikeampi eri polut on ymmärtää, testata, muuttaa ja ylläpitää.

Ohjelmakoodin statistiikka (taulukko 4.4) tuotettiin avoimeen lähdekoodiin perustuvalla työkalulla nimeltä ohcount-tool. Statistiikkaan on laskettu ohjelmointikielikohtaiset lähdekooditiedostojen lukumäärät, niiden koodi- ja kommenttirivit sekä tyhjät rivit.

Taulukko 4.4: Ohjelmakoodin statistiikka

Kieli	Tiedostoja	Koodi	Kommentti	Kommentti-%	Tyhjä	Yhteensä
C++	111	10 506	755	6,7	2 594	13 855
XML	11	1 669	73	4,2	99	1 841
C	8	1 271	696	35,4	464	2 431
Automake	8	212	9	4,1	54	275
Javascript	1	66	31	32,0	4	101
Autoconf	1	39	0	0,0	9	48
Shell	1	6	7	53,8	6	19
Yhteensä	141	13 769	1 571	10,2	3 230	18 570

Syklomaattinen kompleksisuus analysoitiin työkalulla nimeltä pmccabe. Analyysiin käytettiin Carnegie-Mellon Software Engineering Instituten kategorioita [24]:

- 1-10: Yksinkertainen ohjelmistomoduuli ilman merkittävää riskiä.

- 11-20: Keskitason monimutkainen ohjelmistomoduuli keskinkertaisella riskillä.
- 21-50: Monimutkainen ohjelmistomoduuli korkealla riskillä.
- 51 tai suurempi: Testikelvoton ohjelmistomoduuli todella korkealla riskillä.

Allekirjoituspalvelimen ohjelmiston syklomaattinen kompleksisuus on esitetty taulukossa 4.5. McCabe-sarakkeessa on työkalun antama kompleksisuusarvo ja pituus-sarakkeessa funktion pituus ilman kommentteja ja tyhjiä rivejä. Taulukon luettavuuden vuoksi vähäpätöinen ensimmäinen kompleksisuuskategoria (1-10) on poistettu.

Taulukko 4.5: Funktiot, joiden syklomaattinen kompleksisuus yli 10

Funktion nimi	McCabe	Pituus
axis2_svc_skel_SigningService_invoke	21	197
signing::AxisAdapter::init_cert_info	11	52
signing::SigningService::init	15	81
signing::SigningService::sign_request	12	65
signing::SigningService::sign	17	78
signing::SigningService::sign_internal	16	110

Analysoidun ohjelmakoodin ylläpidettävyyden todettiin yleisesti olevan hyvällä tasolla. Yhteensä 141 kooditiedoston joukosta löytyi kolme tiedostoa, jotka sisälsivät yhteensä viisi huomautuksenarvoista funktiota. Taulukon 4.5 funktioista korkeimman kompleksisuusarvon saanut ja ainoa kolmoskategoriaan päätynyt `axis2_svc_skel_SigningService_invoke` sijaitsee axis2-komponentin automaattisesti generoimassa tiedostossa. Tiedoston generointi perustuu projektissa määriteltyihin web-rajapintoihin eikä sitä ei ole tarkoitus käsin ylläpitää. Muut listatut funktiot ovat projektille käsin tuotettua ohjelmakoodia, jotka sijoittuvat kaikki keskitason kategoriaan. Niille suositeltiin tehtäväksi koodikatselmoinnit, mutta lopulta muutoksia ei päädytty tekemään riskien pienuuden vuoksi.

4.5. Virheensieto

Allekirjoituspalvelimen ohjelmiston toteuttaminen sekä Windows- että Linux-käyttöjärjestelmille lisäsi merkittävästi kokonaisjärjestelmän vakautta (engl. robustness), ja tätä kautta allekirjoitustoimintojen saatavuus tietoturvan näkökulmasta kasvoi. Yhdessä versiossa esiintyneet ongelmat, kuten esimerkiksi virheet HSM:ien laiteajureissa, eivät aiheuttaneet ongelmia toisessa versiossa. Lisäksi jos jommankumman

järjestelmän tietoturva- tai jokin muu päivitys aiheutti ongelmia, ei samoja ongelmia ilmennyt toisen järjestelmän kohdalla.

Kun allekirjoituspalvelimissa on ilmennyt ongelmia, joita kuormanjakopalvelimet eivät ole huomanneet (eli ne saivat edelleen vastauksen ping-kutsuun), tarkkailupalvelimen reagointiaika on ollut muutamasta sekunnista kolmeen minuuttiin. Kolme minuuttia on tarkkailupalvelimelle asetettu kutsujen tiheys molemmille tuotannon allekirjoituspalvelimille. Näin on käynyt Windows-pohjaiselle allekirjoituspalvelimelle tuotannossa noin 20 kertaa eikä Linux-pohjaiselle allekirjoituspalvelimelle kertaakaan. Windows-palvelimen ongelmia on aiheuttanut Intelin IT-yksikön automaattisesti ajamat järjestelmäpäivitykset, joita ei ensin olla testattu esituotantoympäristössä. Lisäksi virhe HSM:n laiteajurissa aiheutti ongelmia, kunnes se korjattiin. Allekirjoittajia on tämän vuoksi ohjeistettu asettamaan prosesseihinsa riittävästi joustavuutta, jotta mahdollisen virheen ja hetkellisen katkon sattuessa ne pystyvät automaattisesti jatkamaan heti, kun tarkkailupalvelin on poistanut virheellisen palvelimen kuormanjaosta.

Kuormanjaosta automaattisesti poistetun palvelimen ongelmaan reagoiminen Intelin IT-jaoston järjestelmävalvojien toimesta on yleensä minuutteja. Ongelmien korjausaika on tyypillisesti vaihdellut ongelmasta riippuen muutamasta minuutista kahteen tuntiin. Tänä aikana kaikki allekirjoituskutsut menevät siis kuormanjaossa olevalle palvelimelle.

Hallintapalvelimien käyttökatkot tai toimintahäiriöt eivät ole aiheuttaneet toistaiseksi ainuttakaan ongelmaa allekirjoitusprosesseissa. Tämä johtuu siitä, että allekirjoituspalvelimet ovat kykeneviä toimimaan itsenäisesti paikallisesti tallennettujen konfiguraatioiden ja allekirjoitusavainten avulla.

Isompia ongelmia ja katkoja Intelin tuotanto- ja tuotekehitysprosesseissa syntyisi vasta, jos molemmat allekirjoituspalvelimet olisivat poissa käytöstä. Näin ei ole tuotantokäytössä kertaakaan käynyt ja sen todennäköisyyttä voidaan pitää tilastojen valossa pienenä. Järjestelmään on kuitenkin mahdollista lisätä esimerkiksi kolmas allekirjoituspalvelin pienentämään tätä riskiä, mutta toistaiseksi sen kustannukset on arvioitu saatuun hyötyyn nähden liian korkeiksi.

4.6. Tietoturva-analyysi

Tietoturva-analyysin toteutetusta järjestelmästä tuotti Intelin sisäinen tietoturvalautakunta. Tietoturva-analyysissä arvioidaan, mitä uhkia kohdistuu tiedon luotettavuudelle, eheydelle ja saatavuudelle. Tietoturvasyistä koko analyysia ei tässä työssä yksityiskohtaisesti käydä läpi, mutta yleiset löydökset esitellään korkealla tasolla. Lautakunta sisällytti arviointiinsa kohteet, joiden oletti liittyvän potentiaalisiin hyökkäysvektoreihin ja uhkiin. Nämä kohteet ovat:

- hallintapalvelin
- allekirjoituspalvelin
- HSM
- verkko-komponentit (reitittimet, kuormanjakajat, palomuurit, jne)
- asiakaskirjasto
- älykortit HSM:ien konfigurointiin ja hallintaan.

Koska tietoturva-asiantuntijat olivat luomassa alkuperäisiä vaatimuksia järjestelmälle, olivat lähtökohdat hyväksymisen saamiseen hyvät. Oli kuitenkin asioita, joita analyysissa nousi esille. Nämä on esitetty taulukossa 4.6.

Neljän silmän periaatteen käyttäminen hallinnallisissa prosesseissa tyydytti lautakuntaa. Lisäksi fyysiset järjestelyt olivat heidän mielestään asianmukaisesti toteutettu. Lautakunta antoi järjestelmälle luvan toimia tuotannossa sillä ehdolla, että liiketoiminnasta vastaavat päälliköt hyväksyvät löytyneet haavoittuvuudet ja huomautukset.

4.7. Jatkokehitysmahdollisuuksia

Suoritettujen testien, tehtyjen katselmointien sekä saadun palautteen pohjalta koottiin jatkokehitysideoita, joilla järjestelmää voidaan tarpeen merkittävyyden ja saatavilla olevien resurssien puitteissa parantaa. Nämä jatkokehitysideat eivät sisällä ohjelmointivirheiden korjauksia (engl. bug fixes) tai normaaliin ylläpitoon kuuluvia päivitystoimenpiteitä.

Taulukko 4.6: Tietoturva-analyysin ongelmakohtia

Kohta	Profiili	Huomiot
Käyttäjän todentaminen	sisäpiirihyökkäys; ulkoinen hyökkääjä (kohdennettu hyökkäys, joka vaatii pääsyn Intelin sisäiseen verkkoon)	PKCS#12 on altis brute force -hyökkäyksille eikä riittävää havainnointijärjestelmää varastetun PKCS#12-tiedoston varalle. Järjestelmä ei lisäksi tue politiikassa suositeltua kaksivaiheista tunnistautumista (engl. two-factor authentication).
Allekirjoitettavan tiedon validointi	sisäpiirihyökkäys; ulkoinen hyökkääjä (vaatii sisäpiiriapua)	Vaikka seulontaskriptit suodattavatkin haitalliset allekirjoituspyynnöt oikeista, ei kaikkea tietoa voida seuloa mielekkäästi Lua-skripteillä. Esimerkiksi oikeutettu käyttäjä voi allekirjoittaa haitallista varusohjelmakoodia.

4.7.1. Turvallisuuden parantaminen

Käyttäjän todentaminen oli yksi isoimmista ongelmakohdista tietoturva-analyysissä. Sen parantamiseksi voidaan ajatella ainakin kahta lähestymistapaa:

- Fyysiset elektroniset tunnistautumisvälineet, esimerkiksi älykortit tai usb-kryptotikut.
- Käyttöjärjestelmäkohtaiset ratkaisut integroituun autentikointiin, jotta käyttäjät voivat käyttää Intelin yleisiä käyttäjätunnuksia.

Ensimmäisen vaihtoehdon hyöty on, että se vaatisi hyvin vähän muutoksia nykyiseen ohjelmointiin, sillä nykyinen varmennepohjainen tunnistautuminen voitaisiin korvata esimerkiksi kryptotikulle ladatulla käyttäjän tunnisteella ja yksityisellä avaimella. Tämän vaihtoehdon hankaluus on tikkujen tai korttien provisiointi ja jakaminen insinööreille sekä tähän liittyvien prosessien määrittäminen.

Toinen vaihtoehto vaatii ohjelmallisia muutoksia, mutta lienee käytännöllisempi. Idea olisi käyttää Intelin olemassaolevia Active Directory -palvelimia [25]. Windows-ympäristöissä voitaisiin suoraan käyttää Windowsin omia rajapintoja integroidun autentikoinnin toteuttamiseen ja Linuxilla jotakin siihen soveltuvaa palvelua tai kirjastokokoelmaa (esim. Dellin Authentication Services [26]). Automatisoitujen käännösympäristö-

jen osalta Windowsissa voitaisiin hyödyntää sen natiivia salasanaholvia ja tietokantarajapintoja (DPAPI) [27] ja Linux-ympäristössä esimerkiksi Kerberos-yhteensopivia keytab-tiedostoja [28].

Yhtenä lisäideana yleisen tietoturvan parantamiseen on palvelinten lokitiedostojen allekirjoittaminen nykyisen kahdennuksen lisäksi. Tämä auttaa säilyttämään kiistämättömyyden, vaikka hyökkääjä onnistuisikin lokitiedostojen sisältöä muuttamaan.

4.7.2. Seulonnan parantaminen

Merkittävä puute seulontaskripteissä on se, että kaikki seulontaan käytettävät referenssiarvot joudutaan sisällyttämään vakioina (engl. hard-code) itse skriptiin. Tämän vuoksi referenssiarvon muuttamiseksi sovellusylläpitäjien täytyy päivittää ja ladata koko skripti uudelleen järjestelmään. Tätä voitaisiin parantaa luomalla mahdollisuus asettaa avain-arvo-pareja (engl. key-value pair) allekirjoituskonfiguraatioihin niin, että avain olisi vakion nimi ja parin arvo-osa olisi vakion arvo. Näitä vakioarvoja voitaisiin lukea seulontaskripteistä uuden rajapintafunktion kautta, esimerkiksi `get_definition(KEY_NAME)`. Tämä mahdollistaisi vakioarvojen muuttamisen hallintakäyttöliittymän kautta esimerkiksi SPID-isännän tai uuden tätä varten luodun roolin toimesta. Vakioiden lisäksi skripteillä voisi olla uuden rajapinnan kautta pääsy myös joihinkin oleellisiin muuttujiin kuten esimerkiksi allekirjoituspyynnön mukana toimitettuun metatietoon (komponentti- ja avaintunnisteet sekä allekirjoittavan henkilön tunniste).

Toinen käytettävyyttä parantava ominaisuus olisi mahdollistaa seulontaskriptien asettaminen kokonaisille projekteille, avaimille ja komponenteille. Koska nykyisellään seulontaskriptejä voidaan asettaa vain allekirjoituskonfiguraatioille, on havaittu, että samoja skriptejä käytetään useaan kertaan saman projektin sisällä. Esimerkiksi tiedoston otsakkeen rakenteen tarkastaminen tehdään saman projektin sisällä lähes poikkeuksetta sen kaikille komponenteille. Käytännön syistä olisi helpompaa asettaa kerralla kyseinen skripti ajettavaksi kaikille kyseisen projektin allekirjoituspyynnöille. Tämä vähentäisi myös inhimillisen virheen mahdollisuutta unohtaa skriptin asettaminen yksittäisestä allekirjoituskonfiguraatiosta.

Nykyisen seulontajärjestelmän perimmäinen ongelma on se, että kaikkea tietoa ei ole mielekästä tarkistaa ennaltamääritellyillä Lua-skripteillä. Tämä johtuu kahdesta syystä.

Ensinnäkin nykyisellään skripteistä on hyötyä vain, jos ne suunnitellaan perusteellisesti. Skriptien ohjelmoija voi tehdä tahattomia virheitä skripteihin tai niitä suunnitteleva henkilö voi jättää jotain seulottavia asioita erehdyksessä huomioimatta. Toisekseen skripteillä on käytettävissään hyvin rajallinen määrä metatietoa, vaikka yllä mainitut lisätoiminnot toteutettaisiinkin ja tämän vuoksi seulonnalla on hankala havaita esimerkiksi varastettuja käyttäjätunnuksia.

Ratkaisuna tähän ongelmaan tutkittiin mahdollisuuksia hyödyntää automaattista poikkeavuuksien havainnointijärjestelmää (engl. anomaly detection). Intel tilasi keväällä 2015 Demola-projektin, joka liittyi oleellisesti tähän ideaan. Demola-projektissa tutkittiin ensisijaisesti mahdollisuutta huomata poikkeavuuksia konfiguraatiopakettien allekirjoittamisessa [29]. Näillä paketeilla on hyvin selkeä kenttiin jaettu rakenne, josta voidaan tutkia eri kohtia koneoppimista hyödyntäen. Tämän lisäksi poikkeavuuksia voidaan pyrkiä mahdollisesti tulevaisuudessa tunnistamaan allekirjoituspyyntöön liittyvästä metatiedosta. Kiinnostavia asioita olisi esimerkiksi tietyn allekirjoittajan lähettämän pyynnön poikkeava kellonaika tai maantieteellinen sijainti. Poikkeavuuksien tunnistaminen toimisi hyvänä lisänä seulontasääntöjen rinnalla lisäten järjestelmän luotettavuutta.

4.7.3. Suorituskyvyn parantaminen

Vaikka suorituskyky on jo nykyisellään hyväksyttävällä tasolla, voitaisiin sitä parantaa puuttumalla havaittuun pullonkaulaan eli yhteyden muodostamiseen asiakaslaitteen ja allekirjoituspalvelimen välillä. Esimerkiksi automatisoidun käännösprosessin tapauksessa voidaan tuottaa useita allekirjoitettavia komponentteja. Nykyisellään nämä kaikki lähetetään erikseen omina HTTP-viesteinään, mutta ne voitaisiin paketoida lähetettäväksi yhtenä koottuna pakettina saman yhteyden aikana. Tämä vähentäisi salausoperaatioihin tarvittavaa aikaa, koska se tehtäisiin vain kerran. Muutos kuitenkin monimutkaistaisi myös vastauksen rakennetta, mikä puolestaan vaatisi muutoksia olemassa oleviin allekirjoitustyökaluihin. Muutosten tekeminen rajapintaan ei täten ole yksinkertainen prosessi, vaan vaatii isompaa iterointia työkaluja kehittävien tiimien kanssa.

Suorituskykyä voidaan pitää niin hyvällä tasolla, että sen parantaminen ei ole ensisijaista. Optimointeja voidaan kuitenkin harkita tilanteen mukaan, jos esimerkiksi käyttäjän tunnistautumisen yhteydessä tehdään joka tapauksessa muutoksia web-rajapintoihin.

4.7.4. Lisäominaisuuksia

Mahdollisia lisäominaisuuksia järjestelmään on loputon määrä. Asiakastiimien ja tietoturva-asiantuntijoiden palautteen perusteella uusia varteenotettavia ominaisuuksia voisi kuitenkin olla kaksi.

Ensimmäinen toivottu lisäominaisuus on tuki toimia integroituna osana luotettavaa käännösympäristöä. Pääajatus on, että käyttäjän lähettämän tiedoston sijaan allekirjoituspalvelin osaisi itse noutaa tiedoston luotetulta tiedostopalvelimelta. Tiedostopalvelimelle tullut tiedosto tulisi suoraan käännöspalvelimelta, mikä tekisi sen muuttamisesta ennen allekirjoitusta vaikeampaa. Tämä auttaisi etenkin sellaisten tiedostojen allekirjoittamisessa, joita ei Lua-skripteillä ole mielekästä seuloa, kuten jatkuvasti muuttuva varusohjelmakoodi.

Toinen potentiaalinen lisäominaisuus olisi uuden “todentaja”-roolin luominen. Nykyisessä järjestelmässä julkisen avaimen saamiseksi käyttäjällä pitää olla allekirjoitusoikeudet kyseisen avaimen sisältävään allekirjoituskonfiguraatioon. Tämä ei ole mielekästä sen tahon osalta, jonka vastuulla on toteuttaa allekirjoituksen vahvistaminen loppukohteessa. Tällaisia ovat esimerkiksi insinöörit, jotka polttavat julkisten avainten hajautusarvot laitteiden sulakkeille. Uusi rooli kykenisi noutamaan julkisia avaimia ja niihen liittyvää metatietoa allekirjoituspalvelimilta ilman allekirjoitusoikeuksia.

5. YHTEENVETO

Tämän diplomityön tavoite oli keskittää avainten hallinta ja allekirjoitustoimenpiteet luotettuun palvelinympäristöön. Ensisijaiset asiakkaat olivat Intelin mobiililaitteiden tuotekehityksestä vastaavat tiimit, mutta suunnittelussa haluttiin varautua parhaan mukaan myös muiden liiketoimintayksiköiden allekirjoitustarpeisiin tulevaisuutta ajatellen.

Toteutuksena syntyi digitaalinen allekirjoitusjärjestelmä, jossa avaimet on suojattu HSM-kryptolaitteilla. Allekirjoituspalvelimien konfigurointi tapahtuu hallintapalvelimilla, jotka tarjoavat graafisen web-käyttöliittymän sovellusylläpitäjille. Sekä hallinta- että allekirjoituspalvelimilla on omat HSM-laitteensa. Allekirjoituspalvelimilla olevat avaimet ovat hallintapalvelimilla olevien avainten osajoukko. Käyttäjät lähettävät allekirjoituspyyntönsä allekirjoituspalvelimille käyttäen tuotekehitystiimien omia allekirjoitustyökaluja. Nämä työkalut hyödyntävät samaa dynaamisesti linkitettävää asiakaskirjastoa.

Käyttäjien allekirjoitusoikeudet on sidottu tietomallissa avain-komponentti-sidoksiin, joita nimitetään allekirjoituskonfiguraatioiksi. Näihin konfiguraatioihin voidaan liittää erinäinen määrä seulontaskriptejä, jotka ajetaan allekirjoituspalvelimen toimesta ennen varsinaista allekirjoitusta. Seulontasäännöt määrittävät, minkälaiset tiedostot hyväksytään ja minkälaiset hylätään. Nämä säännöt toteutetaan Lua-kielisillä seulontaskripteillä, jotka pystyvät lukemaan ja kirjoittamaan kenttiä allekirjoitettavasta tiedostosta.

Lopputulos on vakaa tuotantokelpoinen järjestelmä, joka tyydytti asiakastiimien vaatimukset sekä sai Intelin tietoturvalautakunnan ja johdon hyväksynnän toimia tuotannossa. Jatkokehitysajatuksia liittyen sekä tietoturvaan että käytettävyyteen on kuitenkin olemassa. Esimerkiksi seulontajärjestelmää voidaan jatkokehittää luomalla monipuolisempia rajapintoja skriptien käyttöön sekä lisäämällä automaattinen poikkeavuuksien havainnointijärjestelmä tarkkailemaan allekirjoituspyyntöihin liittyvää metatietoa. Tietoturvan osalta merkittävin parannuskohde on käyttäjien tunnistautuminen, joka nyt perustuu käyttäjien henkilökohtaisiin varmenteisiin. Tämä voitaisiin korvata hyödyntämällä Active Directory -palvelimia, mikä on Intelin yleinen käytäntö tietojärjestelmätoteutuksissa.

LÄHTEET

[1] Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista, 2009, saatavilla: http://www.finlex.fi/fi/laki/ajantasa/2009/20090617 [viitattu 11.5.2016]

[2] MTV Uutiset, *Nokia maksoi miljoonia kiristäjälle*, 2014, saatavilla: http://www.mtv.fi/uutiset/rikos/artikkeli/nokia-maksoi-miljoonia-kiristajalle/3448774 [viitattu 11.5.2016]

[3] Diffie, W. ja Hellman, M. E., *New Directions in Cryptography*, IEEE, 1976, saatavilla: https://www-ee.stanford.edu/ hellman/publications/24.pdf [viitattu 12.5.2016]

[4] Rivest, R. L., Shamir, A., Adlemant, L., *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, 1977, saatavilla: http://people.csail.mit.edu/rivest/Rsapaper.pdf [viitattu 11.5.2016]

[5] Kerry, C. F. ja Gallagher, P. D., *Digital Signature Standard (DSS)*, National Institute of Standards and Technology, 2013, saatavilla: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf [viitattu 10.5.2016]

[6] Johnson, D., Menezes, A. ja Vanstone, S., *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, Certicom, 2001 saatavilla: http://cs.ucsb.edu/ koc/ccs130h/notes/ecdsa-cert.pdf [viitattu 10.5.2016]

[7] ElGamal, T., *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE, 1985

[8] National Security Agency, *Commercial National Security Algorithm Suite and Quantum Computing FAQ*, Information Assurance Directorate, 2016, saatavilla: https://cryptome.org/2016/01/CNSA-Suite-and-Quantum-Computing-FAQ.pdf [viitattu 12.5.2016]

[9] Barker, E., Roginsky, A., *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*, National Institute of Standards and Technology, 2015, saatavilla:

http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf [viitattu 11.5.2016]

[10] Dougherty, C., *MD5 vulnerable to collision attacks*, 2008, saatavilla: http://www.kb.cert.org/vuls/id/836068 [viitattu 9.5.2016]

[11] Raggad, B. G., *Information Security Management: Concepts and Practice*, CRC Press, 2010

[12] Reynard, R., *Secret Code Breaker II: A Cryptanalyst's Handbook*, 1997, s. 96

[13] Codenomicon, *The Heartbleed Bug*, saatavilla: http://heartbleed.com/ [viitattu 12.5.2016]

[14] Google Security Blog, *This POODLE bites: exploiting the SSL 3.0 fallback* , 2014, saatavilla: https://security.googleblog.com/2014/10/this-poodle-bites-exploiting-ssl-30.html [viitattu 12.5.2016]

[15] Zhou, Y. B. ja Feng, D. G., *Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing*, Chinese Academy of Sciences, 2005, saatavilla: http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper19.pdf [viitattu 9.5.2016]

[16] Osa, M., *Moorefield Security FAS*, Intelin sisäinen dokumentti, 2015

[17] Intel, *PULSAR GNSS Secure Boot*, Intelin sisäinen dokumentti, 2016

[18] Intel, *Broxton and Apollo Lake Signing and Manifesting Guide*, Intelin sisäinen dokumentti, 2016

[19] The Apache Software Foundation, *Axis2/C*, saatavilla: http://axis.apache.org/axis2/c/core/ [viitattu 9.5.2016]

[20] Skonnard, A., *Understanding SOAP*, Microsoft Developer Network, 2003, saatavilla: https://msdn.microsoft.com/en-us/library/ms995800.aspx [viitattu 17.5.2016]

[21] Gleeson, S. ja Zimman, C., *PKCS #11 Cryptographic Token Interface Base Specification*, Oasis Standard, versio 2.40, 2015, saatavilla: http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.pdf [viitattu 17.5.2016]

[22] Pontifical Catholic University of Rio de Janeiro, *About Lua*, saatavilla: https://www.lua.org/about.html [viitattu 17.5.2016]

[23] Ilvonen, V., *Maintainability Analysis for signing-service*, Intel sisäinen dokumentti, 2013

[24] Munson, J. C., *Software Engineering Measurement*, CRC Press, 2003, s. 332, saatavilla: https://books.google.fi/books?id=FXiRmlJV7-UC [viitattu 11.5.2016]

[25] Microsoft, *Introducing Windows 2000 Server*, luku 11, Microsoft Developer Network, saatavilla: https://msdn.microsoft.com/en-us/library/bb742424.aspx [viitattu 17.5.2016]

[26] Dell Software, *Authentication Services*, saatavilla: http://software.dell.com/products/authentication-services/ [viitattu 17.5.2016]

[27] NAI Labs, *Windows Data Protection*, Microsoft Developer Network, 2001, saatavilla: https://msdn.microsoft.com/en-us/library/ms995355.aspx#windataprotection-dpapi_topic06 [viitattu 17.5.2015]

[28] Massachusetts Institute of Technology, *MIT Kerberos Documentation*, luku "keytab", saatavilla: http://web.mit.edu/Kerberos/krb5-1.13/doc/basic/keytab_def.html [viitattu 17.5.2016]

[29] Niu, Y., Tang, Q., Qian, Y. ja Waris, B., *Anomaly detection in digital signature requests*, Demola Tampere, 2015